

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開2003-223360

(P2003-223360A)

(43)公開日 平成15年8月8日(2003.8.8)

(51)Int.Cl. <sup>7</sup>	識別記号	F I	特許出願公開番号
G 0 6 F 12/08	5 0 7	G 0 6 F 12/08	5 0 7 F 5 B 0 0 5
	5 0 1		5 0 1 C
	5 5 9		5 5 9 E
			5 5 9 Z
	5 6 1		5 6 1

審査請求 未請求 請求項の数7 O L (全 14 頁) 最終頁に続く

(21)出願番号 特願2002-19905(P2002-19905)

(22)出願日 平成14年1月29日(2002.1.29)

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 田原 康宏

東京都小平市上水本町五丁目20番1号 株

式会社日立製作所半導体グループ内

(74)代理人 100080001

弁理士 筒井 大和

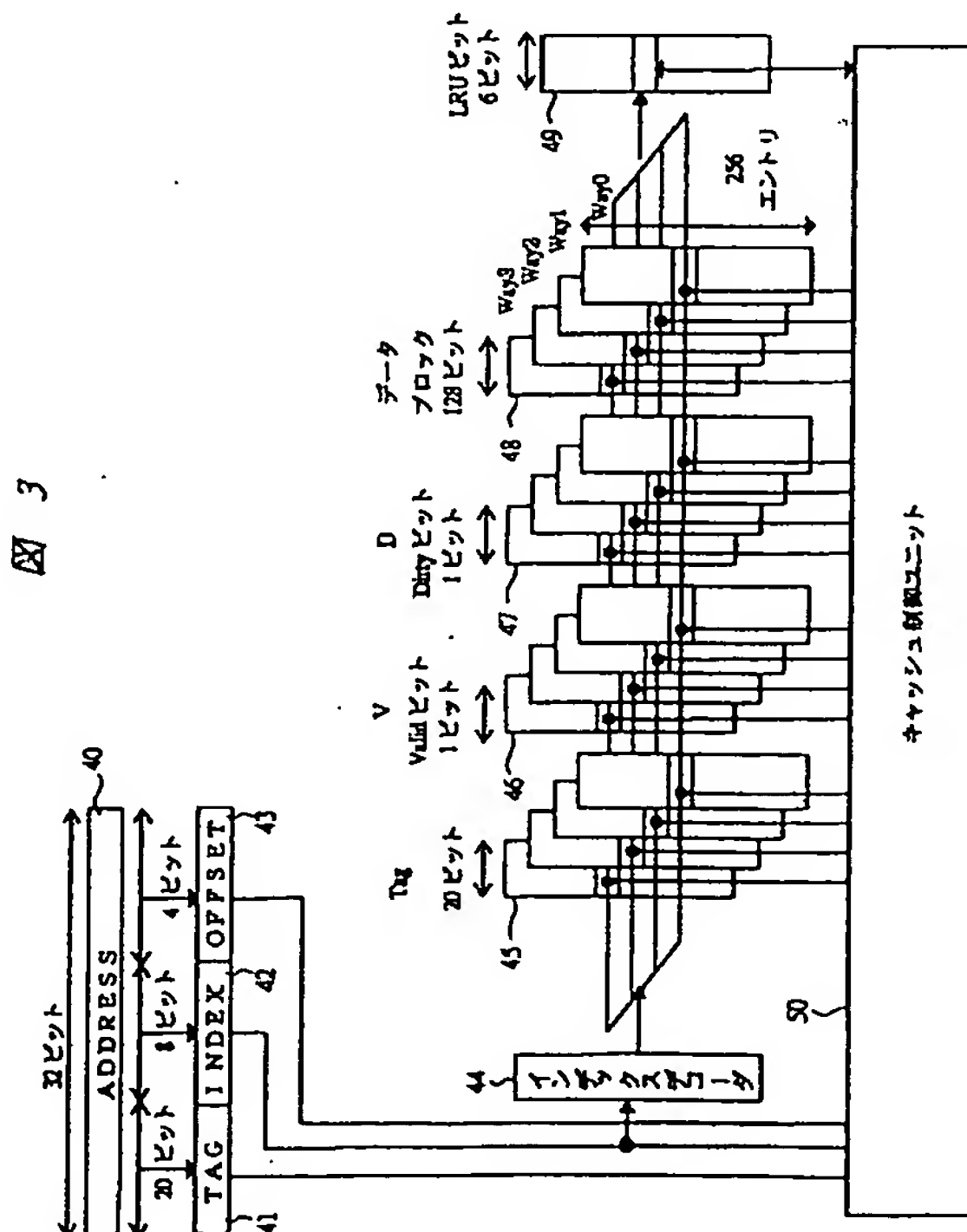
Fターム(参考) 5B005 JJ11 KK12 MM01 NN45 PP03  
QQ05

(54)【発明の名称】 キャッシュメモリシステムおよびマイクロプロセッサ

(57)【要約】

【課題】 主記憶装置とキャッシュメモリとの間の不要なデータ転送を削減して、データ転送に伴う電力消費を削減し、データ転送の経路の混雑を緩和してスループットを向上させることができるキャッシュメモリ制御技術を提供する。

【解決手段】 メインCPUと、ROMとRAMからなる主記憶装置とが外部バスを通じて相互に接続されているキャッシュメモリシステムであって、4-wayセットアソシエイティブキャッシュからなり、各WayはTag 45、Validビット46、Dirtyビット47、データブロック48を持つ。キャッシュエントリをキャッシュから追い出すときに、Dirtyビット47が1にセットされていたらデータブロック48のデータを主記憶装置に書き込み、0にクリアされていたらデータブロック48のデータを主記憶装置に書き込まないで捨ててよい。



BEST AVAILABLE COPY

## 【特許請求の範囲】

【請求項1】 ライトバック方式のキャッシュメモリを有し、  
前記キャッシュメモリ上に確保した領域のダーティビットを操作する命令を実行可能とすることを特徴とするキャッシュメモリシステム。

【請求項2】 請求項1記載のキャッシュメモリシステムにおいて、

前記命令はメモリ解放命令であり、このメモリ解放命令のニモニックとオペランドは、

MREL Rn, IMM

Rn:レジスタ名

IMM:所定のビット数の即値

であることを特徴とするキャッシュメモリシステム。

【請求項3】 請求項1記載のキャッシュメモリシステムにおいて、

前記命令はダーティビットクリア命令であり、このダーティビットクリア命令のニモニックとオペランドは、

DCBDC @Rn

Rn:はレジスタ名

であることを特徴とするキャッシュメモリシステム。

【請求項4】 請求項2または3記載のキャッシュメモリシステムにおいて、

前記Rn:レジスタ名は、スタック領域に動的に確保したメモリ領域を解放するために使用可能なレジスタであることを特徴とするキャッシュメモリシステム。

【請求項5】 請求項2または3記載のキャッシュメモリシステムにおいて、

前記Rn:レジスタ名は、ヒープ領域に動的に確保したメモリ領域を解放するために使用可能なレジスタであることを特徴とするキャッシュメモリシステム。

【請求項6】 請求項1記載のキャッシュメモリシステムにおいて、

前記命令の実行を指示するプロセッサと、主記憶装置とをさらに有し、

前記プロセッサの指示に基づいて前記主記憶装置内の特定のアドレスが指す領域を確保して所定のプログラム処理を行い、

前記確保した領域を前記キャッシュメモリ上に確保して使用した後に、前記プロセッサの指示により前記主記憶装置に確保した領域を解放するとき、前記キャッシュメモリ上に確保した領域のダーティビットをクリアするように制御することを特徴とするキャッシュメモリシステム。

【請求項7】 ライトバック方式のキャッシュメモリを有し、

前記キャッシュメモリ上に確保した領域のダーティビットを操作する命令を実行可能とすることを特徴とするマイクロプロセッサ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、キャッシュメモリ制御技術に関し、たとえばマイクロプロセッサ、マイクロコンピュータ、マイクロコントローラのキャッシュメモリなどを有するキャッシュメモリシステムにおいて、特にダイナミックに確保したメモリ領域を解放するときのキャッシュメモリ制御方式に適用して有効な技術に関する。

【0002】

10 【従来の技術】本発明者が検討したところによれば、キャッシュメモリ制御技術については、たとえば特開平11-338772号、特開平4-268638号、特開平4-264641号、特開平4-188326号、特開平6-28253号、特開平9-231133号の各公報に記載される技術などが挙げられる。

20 【0003】前記公報の技術においては、主記憶装置にダイナミックに確保した領域をキャッシュメモリのエントリとして読み込んだ状態で、主記憶装置の当該領域を解放したときに、当該キャッシュエントリのValidビットをクリアすることにより当該キャッシュエントリを無効にしている。

【0004】

30 【発明が解決しようとする課題】ところで、前記のようなキャッシュメモリ制御技術について、本発明者が検討した結果、以下のようなことが明らかとなった。以下において、本発明者が本発明の前提として検討した技術を、図10～図12を用いて説明する。図10はスタックの使用方法、図11はキャッシュの状態、図12はINDEXが0xA8のキャッシュエントリ、をそれぞれ示す説明図である。

40 【0005】【スタックの一般的使い方の説明：図10】使用前のスタックポインタSPは、SP=0xFEDCBA90という値を持つと仮定する。0xは、次に続く文字列が16進数であることを示す接頭辞である。スタックに0x10バイトの領域を確保するときは、  
SP←SP-0x10  
を実行する。すると、SP=0xFEDCBA80となる。

50 【0006】ユーザ旧SP=0xFEDCBA90と、現SP=0xFEDCBA80との間にある0x10バイトのメモリ領域が使用可能となる。たとえば、  
(SP+0)番地にデータ0x01234567、  
(SP+4)番地にデータ0x89ABCDEF、  
(SP+8)番地にデータ0x01234567、  
(SP+12)番地にデータ0x89ABCDEF、  
を書き込む。これらのデータが不要になり、もはやこれらのデータのためにメモリ領域を確保しておく必要がなくなった時に、

SP←SP+0x10

を実行する。すると、SPは旧SPの値に戻り、確保さ

れた0x10バイトのメモリ領域は解放される。

【0007】スタック用メモリ領域のうち、SPより若いアドレスにあるメモリ領域は未使用で解放されているメモリ領域である。スタック用メモリ領域のうち、SPよりアドレスが大きいメモリ領域は既に確保されているメモリ領域である。

【0008】[ライト時のキャッシュ動作：図11] 図11は、(SP+4)番地、つまり0xFEDCBA84番地にデータ0xFFEEDDCCを書き込んだ後のキャッシュの状態を示している。

【0009】書き込むアドレス0xFEDCBA84がADDRESS60として扱われ、ADDRESSのMSB側から20ビットはTAG61、次の8ビットはINDEX62、最後の4ビットはOFFSET63に分解して扱う。Way J64のINDEX=0xA8のエントリが本書き込み前にInvalid状態(ValidビットV=0)となっていて、Way Jが本書き込みのために選ばれたと仮定する。本エントリでV=0のとき、Tag、Dirtyビット(Dビット)、データブロックには不定値が入っているが、本書き込みによりTagに0xFEDCB、Vビットに1、Dビットに1、データブロックのオフセットがOFFSET=4の位置から4バイトにデータ0xFFEEDDCCが書かれる。

【0010】本説明では、ライトバック方式のキャッシュメモリを仮定しているため、主記憶装置とキャッシュエントリのデータブロックとのコヒーレンシがとれない場合に、Dビットを1にセットする。本データブロックの残りの部分にはライトアロケート機能により、主記憶装置の対応するアドレスからデータをリードする。このときの状態が図11である。

【0011】[本発明の前提技術の具体的な説明：図12] 本発明の前提技術では、

SP←SP+0x10

で、スタックの0x10バイトのメモリ領域を開放すると同時に、解放したメモリ領域が割り付けられているキャッシュエントリのValidビット(Vビット)を0にクリアしてInvalidにしている。仮に、Vビットをクリアしない場合、本キャッシュエントリがLRUアルゴリズムによってキャッシュの外に追い出されるときに、Dビットが1ならばデータブロックの内容が主記憶装置に書き戻される。しかし、本来このデータブロックの内容は既に解放された領域のデータなので、主記憶装置に書き戻しても再度利用されることはなく、書き戻しは無駄である。従って、書き戻しが発生しないようにVビットをクリアすることにより、無用な書き戻しが発生しないようにしていた。当該キャッシュエントリをInvalidにした状態を図12(a)に示す。

【0012】[本発明の前提技術の課題] 前述した本発明の前提技術の課題は、ライトアロケート方式のキャッ

シュメモリにおいて、一度解放した領域と同一のアドレス範囲を再度確保した場合に露呈する。本発明の前提技術では、解放したときにValidビットをクリアするので、当該キャッシュエントリが無効になり、タグやデータを失う。同じアドレス範囲を再度確保して最初に書き込んだときに、タグが失われているので、新たにキャッシュエントリを割り付ける必要がある。ライトアロケート方式に従うと、書き込んだデータを除くキャッシュエントリのデータを主記憶装置からキャッシュエントリに読み込む。このデータ転送により電力を消費したり、データ転送の経路を混雑させて性能が劣化することが、課題として考えられる。この課題を克服することが本発明の目的である。

【0013】たとえば、0xFEDCBA80から始まる16バイトのスタック領域を解放したときに、本発明の前提技術により図12(a)のように当該キャッシュエントリをInvalidにしたと仮定する。次に、0xFEDCBA84番地に0BBBBBAAAという値を書き込み、同じWayにアロケートされたと仮定すると、図12(b)の状態になる。このとき、当該データブロックのライトした4バイトデータを除く部分は主記憶装置から読み込まれる。この読み込みにより消費される電力を削減することと、この読み込みによるデータを転送する経路の混雑をなくし、スループットを向上することが本発明の目的である。

【0014】すなわち、本発明の目的は、主記憶装置とキャッシュメモリとの間の不要なデータ転送を削減して、データ転送に伴う電力消費を削減し、データ転送の経路の混雑を緩和してスループットを向上させることができるキャッシュメモリ制御技術を提供することにある。

【0015】本発明の前記ならびにその他の目的と新規な特徴は、本明細書の記述および添付図面から明らかになるであろう。

【0016】

【課題を解決するための手段】本願において開示される発明のうち、代表的なものの概要を簡単に説明すれば、次のとおりである。

【0017】本発明によるキャッシュメモリシステムは、ライトバック方式のキャッシュメモリを有し、このキャッシュメモリ上に確保した領域のダーティビットを操作する命令を実行可能とするものである。特に、前記命令は、メモリ解放命令MREL Rn、IMM、あるいはダーティビットクリア命令DCBDC @Rn、であり、スタック領域、あるいはヒープ領域に動的に確保したメモリ領域を解放するために使用可能なレジスタに適用するものである。

【0018】さらに、前記キャッシュメモリシステムにおいて、命令の実行を指示するプロセッサと、主記憶装置とを有し、プロセッサの指示に基づいて主記憶装置内



の特定のアドレスが指す領域を確保して所定のプログラム処理を行い、この確保した領域をキャッシュメモリ上に確保して使用した後に、プロセッサの指示により主記憶装置に確保した領域を解放するとき、キャッシュメモリ上に確保した領域のダーティビットをクリアするように制御するものである。

【0019】すなわち、本発明は、本発明の前提技術において、主記憶装置の解放した領域に対応するキャッシュエントリのValidビットをクリアしていたのに対して、本発明では当該キャッシュエントリのDirty 10 ビットをクリアする点を特徴とするものである。

【0020】これにより、Dirtyビットをクリアしたキャッシュエントリがキャッシュに残っている場合に、同一のアドレス範囲にある主記憶装置の領域を新たに確保し、この領域に最初にデータを書き込んだ時に本発明の効果が現れる。既に当該アドレス範囲に対応するキャッシュエントリがキャッシュに存在するため、ライトアロケートする必要がなく、データ転送が発生しない。よって、データ転送のための電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることも 20 ない。

【0021】また、DirtyビットをクリアしたキャッシュエントリがLRUアルゴリズムなどによりキャッシュから追い出される場合においても、Dirtyビットがクリアされているために、キャッシュエントリのデータを主記憶装置に書き戻す必要がなく、書き戻しのためのデータ転送が発生しない。よって、データ転送のための電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることもない。

【0022】たとえば、0xFEDCBA80から始まる16バイトのスタック領域が、前述した図11のようにWay JのINDEX=0xA8のキャッシュエントリに割り付けられている状態で、本領域を解放したときに、本発明では図12(c)のように当該キャッシュエントリのDirtyビットを0にクリアする。このキャッシュエントリがキャッシュの外に追い出されないうちに0xFEDCBA84番地に0BBBBBAAAという値を書き込むと、図12(d)の状態になる。このとき、当該データブロックのライトしたデータを除く 40 部分は主記憶装置から読み込まれることはなく、データブロックに元々書かれていた値である。従って、本発明の前提技術ではライトアロケート方式により主記憶装置からの読み込みが発生していたが、本方式ではこの読み込みが発生しない。この読み込みにより消費される電力を削減され、この読み込みによるデータを転送する経路の混雑は発生せず、スループットを向上することができる。

【0023】また、図12(c)の状態のキャッシュエントリがLRUアルゴリズムによりキャッシュの外に追い出されるとき、Dirtyビットが0にクリアされて 50

いるため、書き戻しのためのデータ転送の電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることもない。

【0024】

【発明の実施の形態】以下、本発明の実施の形態を図面に基づいて詳細に説明する。

【0025】まず、図1により、本発明を応用した本発明の一実施の形態の携帯電話システムの構成の一例を説明する。図1は本実施の形態の携帯電話システムを示す説明図であり、(a)は平面図、(b)は機能ブロック図をそれぞれ示す。

【0026】本実施の形態の携帯電話システムは、無線信号を送受信する無線部101、送受信信号を変調/復調処理するベースバンド回路102、信号をフィルタ処理するDSP103、アナログ/デジタル変換するA/D変換器104、信号増幅するAF回路105、音声を出力するスピーカ106、音声を入力するマイク107からなる電話機能部と、表示信号を演算処理する操作部・CPU108、表示用のLCD109、LCD109を駆動するLCDドライバ110、キー入力するキー入力部111からなる表示機能部と、全体の演算処理を司るメインCPU112、主記憶装置のROM113、RAM114、フラッシュメモリ115からなる制御機能部などから構成されている。

【0027】電話機能部における受信時は、無線信号を無線部101で受信すると、この無線信号がベースバンド回路102により復調処理され、さらにDSP103によりフィルタ処理された後に、A/D変換器104によりアナログ信号からデジタル信号に変換される。そして、デジタル信号は、AF回路105により増幅され、スピーカ106を通じて音声信号として出力される。

【0028】さらに、送信時は、マイク107から音声を入力すると、音声信号がAF回路105により増幅され、さらにA/D変換器104によりデジタル信号からアナログ信号に変換される。そして、DSP103によりフィルタ処理され、ベースバンド回路102により変調処理された後に、無線部101を通じて無線信号として送信される。

【0029】また、表示機能部においては、電話の情報や付加的に設けられた電子メールなどの各種情報が操作部・CPU108により演算処理され、これらの各種情報がLCDドライバ110を介してLCD109に表示される。また、キー入力部111からの入力により、各種機能の選択や、電子メールの文字入力などを行うことができる。

【0030】また、制御機能部においては、携帯電話システムの全体の演算処理がメインCPU112で実行され、このメインCPU112による各種演算処理は、たとえばROM113やフラッシュメモリ115に記憶されている各種プログラムに基づいて行われ、これらの各

種演算処理のデータは随時、たとえばRAM114に格納される。この制御機能部のメインCPU112、ROM113、RAM114からなる部分についての詳細は後述する。

【0031】次に、図2により、前記図1のメインCPUとROMとRAMからなる部分のキャッシュメモリシステムの構成の一例を説明する。図2はメインCPUとROMとRAMからなる部分のキャッシュメモリシステムを示すブロック図である。

【0032】図2において、キャッシュメモリシステムは、プロセッサであるメインCPU1と、ROMとRAMからなる主記憶装置2とは外部バス3を通じて相互に接続されている。なお、図2におけるメインCPU1は図1のメインCPU112に対応し、また図2における主記憶装置2のROMとRAMは図1のROM113とRAM114にそれぞれ対応する。

【0033】メインCPU1には、データキャッシュ10、命令キャッシュ11、ライトバッファ12、制御ユニット13、命令バッファ14、命令デコードユニット15、レジスタファイル16、スタックポインタ6、プログラムカウンタ17、演算回路18、メモリデータアクセスユニット19、ライトバックユニット20、バスユニット21が設けられ、これらの各ユニットは内部アドレスバス22、内部データバス23に任意に接続されている。また、このメインCPU1のバスユニット21は、外部アドレスバス24、外部データバス25、外部制御信号線26を通じて、外部バス3、主記憶装置2に接続されている。

【0034】主記憶装置2には、ROM4、RAM5が設けられている。ROM4には、メインCPU1を制御するプログラムと定数データが書き込まれている。RAM5には、静的に確保されるメモリ領域と、ヒープ領域とスタック領域のように動的に確保されるメモリ領域がある。

【0035】続いて、メインCPU1の動作を説明する。プログラムカウンタ17は命令アドレスを保持し、この命令アドレスにある命令を命令キャッシュ11から命令バッファ14に送る。このとき、命令キャッシュ11は命令アドレスの内容を保持していない場合は、内部アドレスバス22と内部データバス23、バスユニット21、外部アドレスバス24と外部データバス25と外部制御信号線26、外部バス3を経由して主記憶装置2にあるROM4から命令を読み込み、所定の形式で保持する。

【0036】命令バッファ14から命令を命令デコードユニット15に転送し、命令デコード結果が制御ユニット13を介して、レジスタファイル16から必要ならばデータを読み出し、演算回路18、メモリデータアクセスユニット19、ライトバックユニット20からなるデータバスを通り処理される。

【0037】メモリデータアクセスユニット19はデータキャッシュ10に対してアドレスを生成し、当該アドレスのデータ読み出しをデータキャッシュ10に要求し、データキャッシュ10は所定の処理を行い、要求されたデータをメモリデータアクセスユニット19に出力する。

【0038】ライトバックユニット20は、レジスタファイル16、または、データキャッシュ10にデータを出力する。出力先がデータキャッシュ10の場合は、書き込むデータ、および、書き込み先アドレスをデータキャッシュ10に出力する。

【0039】[データの読み出し] データキャッシュ10に当該アドレスのデータがある場合は、データキャッシュ10が要求を処理する。データキャッシュ10に当該アドレスのデータがない場合、当該アドレスは内部アドレスバス22を経由してバスユニット21に転送される。バスユニット21は、外部アドレスバス24と外部制御信号線26に信号を出力して、外部バス3に要求を出し、主記憶装置2にあるROM4またはRAM5からデータを読み出す。読み出したデータは、外部制御信号線26の信号により外部バス3から外部データバス25を経由してバスユニット21に転送され、内部アドレスバス22を経由してデータキャッシュ10に読み込まれる。データキャッシュ10は所定の形式でデータを保持する。

【0040】[データの書き込み] データキャッシュ10に書き込み先アドレスのデータが保持されている場合は、データキャッシュ10の所定の場所にデータを書き込む。ここでは、ライトアロケート方式のキャッシュを仮定している。データキャッシュ10に当該アドレスのデータが保持されていない場合、データキャッシュ10にデータを保持する場所を確保し、データを書き込む。確保した場所のサイズが書き込んだデータのサイズより大きい場合は、書き込んだデータを除く部分を主記憶装置2から読み込む。

【0041】ここでは、ライトバック方式のキャッシュを仮定している。データキャッシュ10が主記憶装置2に書き込みを要求する場合は、当該アドレスとデータは、ライトバッファ12に一時的に保持され、それぞれ内部アドレスバス22と内部データバス23を経由し、バスユニット21に転送される。バスユニット21は、外部アドレスバス24と外部データバス25と外部制御信号線26に信号を出力して、外部バス3に要求を出し、主記憶装置2にあるRAM5に対してデータを書き込む。

【0042】本実施の形態では、命令キャッシュ11とデータキャッシュ10が分離されている例を示しているが、命令キャッシュ11とデータキャッシュ10を融合したキャッシュにおいても、本発明を実施することができ

【0043】また、データキャッシュ10と主記憶装置2の間に二次キャッシュがある場合、データキャッシュおよび二次キャッシュにおいても、本発明を実施することが可能である。

【0044】次に、図3により、キャッシュの構成の一例を説明する。図3はキャッシュの構成を示す説明図である。

【0045】本例は、4-wayセットアソシエティブキャッシュである。Way0, Way1, Way2, Way3の4つのWayからなり、各WayはTag 45、Validビット(Vビット) 46、Dirtyビット(Dビット) 47、データブロック48を持つ。

【0046】1つのWayは256エン트리からなり、各エントリは20ビットのTag 45、1ビットのValidビット46、1ビットのDirtyビット47、128ビットのデータブロック48からなる。

【0047】ADDRESS 40は、リードまたはライトを要求されているアドレスを示す。本例では32ビットである。TAG 41は、本例ではADDRESS 40のMSB側の20ビットである。OFFSET 43は、本例ではADDRESS 40のLSB側の4ビットである。INDEX 42は、TAG 41とOFFSET 43の間にある8ビットである。

【0048】INDEX 42は、インデックスデコーダ44により、各Wayの256個のエントリのINDEX番目の1つを指す。TAG 41は、キャッシュエントリにあるTag 45に保持される。データブロック48のアドレスは、キャッシュエントリにあるTag 45とINDEXから復元することができる。

【0049】Validビット46が1にセットされている場合、当該エントリが有効であり、0にクリアされている場合は無効である。

【0050】Dirtyビット47が1にセットされている場合、データブロック48の内容が書き込みにより、主記憶装置の対応するメモリブロックより新しいデータに更新されていることを示す。Dirtyビット47が0にクリアされている場合は、一般には主記憶装置の対応するメモリブロックと同じデータを持つことを示す。キャッシュエントリをキャッシュから追い出すときに、Dirtyビット47が1にセットされていたらデータブロック48のデータを主記憶装置に書き込み、0にクリアされていたらデータブロック48のデータを主記憶装置に書き込まないで捨ててよい。

【0051】LRUビット49で、Way数をnとすると各INDEX番目のLRUビットは、 $n \times (n-1) / 2$ 個のビット数からなる。本例は4wayなので、LRUビット49は各INDEXで、 $4 \times 3 / 2 = 6$ により6ビットからなる。つまり、LRUビット49は

4つのWayで同じインデックス番号を持つエントリの4つの中から選ぶペアの組み合わせは6ペアあり、各ペアに関して新旧関係をビットで表現している。6つのビットの値からLeast-Recently-Used (LRU)のエントリを見つけることができる。

【0052】キャッシュ制御ユニット50による、リードの場合の動作を図4に示し、ライトの場合の動作を図5に示す。このキャッシュ制御ユニット50は、たとえば前記図2のデータキャッシュ10の一部として構成される。図4はキャッシュのリード動作を示すフロー図、図5はキャッシュのライト動作を示すフロー図である。

【0053】キャッシュのリード動作は、図4に示すように、まず開始後(ステップS100)、ステップS101において、TAGにADDRESS[31:12]、INDEXにADDRESS[11:4]、OFFSETにADDRESS[3:0]をそれぞれ代入する。

【0054】さらに、ステップS102において、各i(i=0, 1, ..., N-1)で第i番目のWayのINDEX番目のエントリに対して、Valid(i)=1でTag==TAGのとき、1→Hit(i)、i→J、Valid(i)=0、あるいはValid(i)=1でTag≠TAGでないとき、0→Hit(i)、の論理条件によりHit(i)を求める。そして、ステップS103において、求めたHit(0), Hit(1), ..., Hit(N-1)を論理和演算し、HITに代入する。

【0055】さらに、ステップS104において、HITを判別し、HIT=1、すなわち1つでもヒットしたときは、ステップS105において、第J番目のWayのINDEX番目のエントリのデータブロックのOFFSETにより指されているアドレスからアクセスサイズのデータを読み込む。本ブロックをmost-recently-usedにLRUフィールドをアップデートする。

【0056】また、HIT=0、すなわち1つもヒットしないときは、ステップS106において、0からN-1のWayのINDEX番目のエントリのうち、あるK番目のWayでV(K)=0を判別し、Noのときは、ステップS107において、全WayのINDEX番目のエントリからLRUブロックを選択し、これが第K番目のWayにあると仮定する。

【0057】さらに、ステップS108において、D(K)を判別し、D(K)=1のときは、ステップS109において、第K番目のWayのINDEX番目のエントリのTag(K)とデータブロックおよびINDEXをライトバッファに待避する。当該エントリのデータブロックのOFFSETに対応するワードからラップアラウンド方式で外部メモリからデータを読み込み、該当するデータがキャッシュへ到達した時点でCPUへ読み



出しデータを返す。本ブロックをmost-recently-usedにLRUフィールドをアップデートする。当該エントリで、TAG→Tag(K)、0→D(K)、1→V(K)のそれぞれの代入を行い、ライトバッファのデータを待避してあるTag(K)およびINDEXが指すアドレスに書き込む。

【0058】また、ステップS108においてD(K)=0、あるいはステップS106におけるV(K)==0の判別の結果がYesのときは、ステップS110において、第K番目のWayのINDEX番目のエントリのデータブロックのOFFSETに対応するワードからラップアラウンド方式で外部メモリからデータを読み込み、該当するデータがキャッシュへ到達した時点でCPUへ読み出しデータを返す。本ブロックをmost-recently-usedにLRUフィールドをアップデートする。当該エントリで、TAG→Tag(K)、1→V(K)、0→D(K)のそれぞれの代入を行う。

【0059】キャッシュのライト動作は、図5に示すように、開始後(ステップS200)、前記リード動作と同様に、ステップS201～S203において処理した後、ステップS204において、HITを判別し、HIT=1のとき、ステップS206でV(K)==0となるWay Kが存在せずステップS207でWay KがLRUとなりステップS208でD(K)=1のとき、D(K)=0(ステップS208)、あるいはステップS206であるWay Kに対しV(K)==0の判別の結果がYesのときは、それぞれ以下のようになる。

【0060】HIT=1のときは、ステップS205において、第J番目のWayのINDEX番目のエントリのデータブロックのOFFSETにより指されているアドレスにアクセスサイズのデータを書き込む。本ブロックをmost-recently-usedにLRUフィールドをアップデートする。1→D(J)の代入を行う。

【0061】ステップS208でD(K)=1のときは、ステップS209において、第K番目のWayのINDEX番目のエントリのTag(K)とデータブロックおよびINDEXをライトバッファに待避する。当該エントリのデータブロックのOFFSETが指すアドレスにアクセスサイズのデータを書き込み、データブロックの残りの部分へ外部メモリからデータを読み込む。本ブロックをmost-recently-usedにLRUフィールドをアップデートする。当該エントリで、TAG→Tag(K)、1→D(K)、1→V(K)のそれぞれの代入を行い、ライトバッファのデータを待避してあるTag(K)およびINDEXが指すアドレスに書き込む。

【0062】ステップS208でD(K)=0、あるいはステップS206であるWay Kに対しV(K)==0の判別の結果がYesのときは、ステップS210に

において、第K番目のWayのINDEX番目のエントリのデータブロックのOFFSETが指すアドレスにアクセスサイズのデータを書き込み、データブロックの残りの部分へ外部メモリからデータを読み込む。本ブロックをmost-recently-usedにLRUフィールドをアップデートする。当該エントリで、TAG→Tag(K)、1→D(K)、1→V(K)のそれぞれの代入を行う。

【0063】本発明は、ライトバック方式かつライトアロケート方式のキャッシュで効果がある。以下において、ライトバック方式、ライトアロケート方式を説明する。

【0064】ライトバック方式は、ライトスルー方式と対になるキャッシュ制御方式である。ライトスルー方式では、あるアドレスにライトするときに、キャッシュに当該アドレスのエントリがあれば、そのデータブロックにライトするとともに主記憶装置の当該アドレスにもライトする。この場合、エントリのデータブロックの内容が主記憶装置の当該メモリブロックの内容と一致するので、Dirtyビットを1にセットする必要はない。

【0065】ライトバック方式では、あるアドレスにライトするときに、キャッシュに当該アドレスのエントリがあれば、そのデータブロックにライトするが、主記憶装置にはライトしない。この時、Dirtyビットに1をセットしてデータブロック内容が主記憶装置の当該メモリブロックの内容と一致しないことを表す。

【0066】ライトアロケート方式は、あるアドレスのライトにより当該アドレスが属するメモリブロックをキャッシュエントリに割り付ける方式である。リードした場合は、通常キャッシュエントリに割り付けるが、ライトの場合はライトアロケートかライトスルーかの選択の余地がある。ライトアロケート方式でライトの当該アドレスをキャッシュエントリに割り付けた場合、ライトした部分ではないデータを主記憶装置からキャッシュのデータブロックに読み込む。

【0067】【第1の実施の形態】本実施の形態を、図6、図7および図8により説明する。図6は動的に確保される領域のキャッシュ制御方式を示すフロー図、図7は図6と図8とで使用している変数の説明図、図8はメモリ解放命令の動作を示すフロー図である。

【0068】本実施の形態では、前述したデータキャッシュをインプリメントしたCPUを考える。ライトバック方式かつライトアロケート方式とを仮定している。

【0069】本実施の形態は、図6の動的に確保された領域のキャッシュ制御方式のアルゴリズムを実現するためのものである。本実施の形態の例では図6のステップS301の判別はコンパイラまたはプログラマにより既になされているものとする。また、図8のステップS409に示すようにスタック解放に使用できるようにレジスタの値を解放したサイズだけ加算する機能を付け加え

ている。

【0070】すなわち、図6において、動的に確保された領域のキャッシュ制御方式は、まず開始後（ステップS300）、ステップS301の動的に確保したか否かの判別処理において、動的に確保したメモリ領域を解放したか否かを判別し、解放していないとき（No）は終了（ステップS306）となり、解放している場合（Yes）は、ステップS302以降の処理に進む。

【0071】ステップS302の初期化処理において、変数S、R、E、B、MASK、SS、EEに対し、解放するメモリ領域の先頭アドレス→S、解放するメモリのバイトサイズ→R、S+R（解放するメモリ領域の終了アドレス）→E、キャッシュブロックのバイトサイズ→B、NOT (B-1) →MASK、S AND MASK + {B if {S AND (B-1)} ≠ 0, 0 if {S AND (B-1)} = 0} →SS、S AND MASK →EE、のそれぞれの代入を行う。ここでNOT xは32ビット幅でxのビット毎の反転を表す。x AND yは32ビット幅でxとyとのビット毎の論理積を表す。

【0072】さらに、ステップS303のループ終了判別処理において、SS < EEを判別し、Noのときは終了となり、Yesの場合は、ステップS304のDirtyビットクリア処理において、SS番地に対応するキャッシュエントリのDirtyビットをクリアし、そしてステップS305のアドレスカウンタアップデート処理において、SS+B→SSの代入を行った後、ステップS303からの処理を繰り返す。

【0073】ここで、図7により、前述したS、R、E、B、MASK、SS、EEの変数を具体的に説明する。

【0074】図7においては、たとえば、解放するメモリ領域の先頭アドレスに対応する変数S = 0x100C、解放するメモリのバイトサイズに対応する変数R = 0x38、解放するメモリ領域の終了アドレスに対応する変数E = S + R = 0x100C + 0x38 = 0x1044、キャッシュブロックのバイトサイズに対応する変数B = 16、とした場合に、補正の変数MASK = NOT (B-1) = NOT (0xF) = 0xFFFFF0となり、補正後の変数SS = 0x1010、変数E = 0x1040となる。

【0075】言い換えれば、SSはS（解放するメモリ領域の先頭アドレス = 0x100C）をB（キャッシュブロックのバイトサイズ = 16）の倍数に切り上げた数、EEはE（解放するメモリ領域の終了アドレス = 0x1044）をBの倍数に切り捨てた数となる。すなわち、図7において、破線のメモリ領域は他で使っているかもしれないので除外する必要がある。

【0076】本実施の形態では、スタック領域の解放をサポートする命令として、本発明を採用している。この

メモリ解放命令のニモニックとオペランドを以下に示す。

【0077】MREL Rn, IMM

Rn: レジスタ名

IMM: 所定のビット数の即値

MREL命令は、図8のメモリ解放命令MREL Rn, IMMの動作のアルゴリズムに表した動作を行う。

【0078】今、スタック領域を関数からリターンする直前に解放する場合を考える。スタックポインタをR15、解放したいメモリサイズを32バイトとすると、MREL R15, 32

により、R15 ← R15 + 32を実行して、スタックの32バイトの領域を解放するとともに、変更前のR15と変更後のR15で挟まれた領域のキャッシュエントリのDirtyビットがクリアされる。Dirtyビットをクリアするときは、図8のステップS401とS402に書かれているように、解放の対象となっていないアドレスを含むキャッシュエントリは解放の対象外とするように解放の範囲を計算している。

【0079】すなわち、図8において、メモリ解放命令MREL Rn, IMMの動作は、まず開始後（ステップS400）、ステップS401の初期化処理において、変数S、R、E、B、MASK、SS、EEに対し、解放するメモリ領域の先頭アドレスRn → S、解放するメモリのバイトサイズIMM → R、解放するメモリ領域の終了アドレス (S + R) → E、キャッシュブロックのバイトサイズ → B、NOT (B - 1) → MASK、S AND MASK + {B if {S AND (B - 1)} ≠ 0, 0 if {S AND (B - 1)} = 0} → SS、E AND MASK → EE、のそれぞれの代入を行う。

【0080】さらに、ステップS402のループ終了判別処理において、SS < EEを判別し、Noのときは、ステップS409のレジスタアップデート処理において、Rn + IMM → Rn、の代入を行って終了（ステップS410）となり、Yesの場合は、ステップS403以降の処理に進む。

【0081】ステップS403のアドレス分解処理において、TAGにSS[31:12]、INDEXにSS[11:4]、OFFSETにSS[3:0]をそれぞれ代入する。さらに、ステップS404のキャッシュヒット判別処理において、各i (i = 0, 1, ..., N - 1) で第i番目のWayのINDEX番目のエントリに対して、Valid(i) = 1でTag == TAGのとき、1 → Hit(i)、i → J、Valid(i) = 0、あるいはValid(i) = 1でTag == TAGでないとき、0 → Hit(i)、の論理条件によりHit(i)を求める。そして、ステップS405のキャッシュヒット集計処理において、求めたHit(0), Hit(1), ..., Hit(N - 1)を論理和演算し、H



ITに代入する。

【0082】さらに、ステップS406のキャッシュヒット判別処理において、HITを判別し、HIT=1、すなわち1つでもヒットしたときは、ステップS407のDirtyビットクリア処理において、第J番目のWayのINDEX番目のエントリに対し0→Dirty(J)の代入を行った後、ステップS408に進む。

【0083】また、HIT=0、すなわち1つもヒットしないときは、ステップS408のアドレスカウンタアップデート処理において、SS+B→SSの代入を行った後に、ステップS402からの処理を繰り返して実行する。

【0084】なお、マルチタスク方式でスタック領域がタスク毎に排他的に割り付けられている場合も、本実施の形態の例をそのまま適用できる。

【0085】〔第2の実施の形態〕本実施の形態は、前記第1の実施の形態の変形例である。前記第1の実施の形態で示した、MREL命令に指定するレジスタはスタックポインタに使われているレジスタだけとは限らない。ヒープ領域に動的に確保したメモリ領域を解放するときに使うことができる。

【0086】具体的には、C言語の標準ライブラリ関数にあるfree関数の内部で解放した領域に対してMREL命令を施す例がある。解放する領域の先頭アドレスをR1、解放するバイト数を100とすると、

MREL R1, 100  
を実行することにより、解放した領域に対応するキャッシュエントリのダーティビットがクリアされる。

【0087】〔第3の実施の形態〕本実施の形態を、図9により説明する。図9はDirtyビットクリア命令の動作を示すフロー図である。

【0088】本実施の形態では、前記図6のステップS304をCPUの命令として実現したものである。Dirtyビットクリア命令のオペランドを以下に示す。

【0089】DCBDC @Rn

Rn：レジスタ名

DCBDC命令は、図9のDirtyビットクリア命令DCBDC @Rnの動作のアルゴリズムに表した動作を行う。

【0090】すなわち、図9において、Dirtyビットクリア命令DCBDC @Rnの動作は、まず開始後（ステップS500）、ステップS501の対象アドレス取得処理において、SSにRn（アドレス）を代入する。

【0091】さらに、ステップS502のアドレス分解処理において、TAGにSS[31:12]、INDEXにSS[11:4]、OFFSETにSS[3:0]をそれぞれ代入する。さらに、ステップS503のキャッシュヒット判別処理において、各i (i=0, 1,

..., N-1)で第i番目のWayのINDEX番目のエントリに対して、Valid(i)=1でTag==TAGのとき、1→Hit(i)、i→J、Valid(i)=0、あるいはValid(i)=1でTag==TAGでないとき、0→Hit(i)、の論理条件によりHit(i)を求める。そして、ステップS504のキャッシュヒット集計処理において、求めたHit(0), Hit(1), ..., Hit(N-1)を論理和演算し、HITに代入する。

【0092】さらに、ステップS505のキャッシュヒット判別処理において、HITを判別し、HIT=1、すなわち1つでもヒットしたときは、ステップS506のDirtyビットクリア処理において、第J番目のWayのINDEX番目のエントリに対し0→Dirty(J)の代入を行って終了（ステップS507）となる。また、HIT=0、すなわち1つもヒットしないときは、終了となる。

【0093】従って、前記実施の形態によれば、主記憶装置の解放した領域に対応するキャッシュエントリのDirtyビットをクリアすることにより、このDirtyビットをクリアしたキャッシュエントリがキャッシュに残っている場合に、同一のアドレス範囲にある主記憶装置の領域を新たに確保し、この領域に最初にデータを書き込んだ時に、既に当該アドレス範囲に対応するキャッシュエントリがキャッシュに存在するため、ライトアロケートする必要がなく、データ転送が発生しない。よって、データ転送のための電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることもない。

【0094】また、DirtyビットをクリアしたキャッシュエントリがLRUアルゴリズムなどによりキャッシュから追い出される場合においても、Dirtyビットがクリアされているために、キャッシュエントリのデータを主記憶装置に書き戻す必要がなく、書き戻しのためのデータ転送が発生しない。よって、データ転送のための電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることもない。

【0095】以上、本発明者によってなされた発明をその実施の形態に基づき具体的に説明したが、本発明は前記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることはいうまでもない。

【0096】

【発明の効果】本願において開示される発明のうち、代表的なものによって得られる効果を簡単に説明すれば、以下のとおりである。

【0097】(1) ダイナミックに確保した主記憶装置の領域を解放するときに、対応するキャッシュエントリのDirtyビットをクリアすることで、再度同一領域を確保した場合に、同一キャッシュエントリにライトアロケートが発生しないので、ライトアロケートによるキ

キャッシュメモリと主記憶装置間のデータ転送をなくすことが可能となる。

【0098】(2)前記(1)により、主記憶装置とキャッシュメモリとの間の不要なデータ転送を削減することができるので、このデータ転送に伴う電力消費を削減し、データ転送の経路の混雑を緩和してスループットを向上させることが可能となる。

【図面の簡単な説明】

【図1】(a)、(b)は本発明の一実施の形態の携帯電話システムを示す説明図である。

【図2】本発明の一実施の形態において、メインCPUとROMとRAMからなる部分のキャッシュメモリシステムを示すブロック図である。

【図3】本発明の一実施の形態において、キャッシュの構成を示す説明図である。

【図4】本発明の一実施の形態において、キャッシュのリード動作を示すフロー図である。

【図5】本発明の一実施の形態において、キャッシュのライト動作を示すフロー図である。

【図6】本発明の一実施の形態において、動的に確保される領域のキャッシュ制御方式を示すフロー図である。

【図7】本発明の一実施の形態において、変数の説明図である。

【図8】本発明の一実施の形態において、メモリ解放命令の動作を示すフロー図である。

【図9】本発明の一実施の形態において、Dirtyビットクリア命令の動作を示すフロー図である。

【図10】本発明の前提として検討した技術において、スタックの使用法を示す説明図である。

【図11】本発明の前提として検討した技術において、キャッシュの状態を示す説明図である。

【図12】(a)～(d)は本発明の前提として検討した技術および本発明の技術において、INDEXが0xA8のキャッシュエントリを示す説明図である。

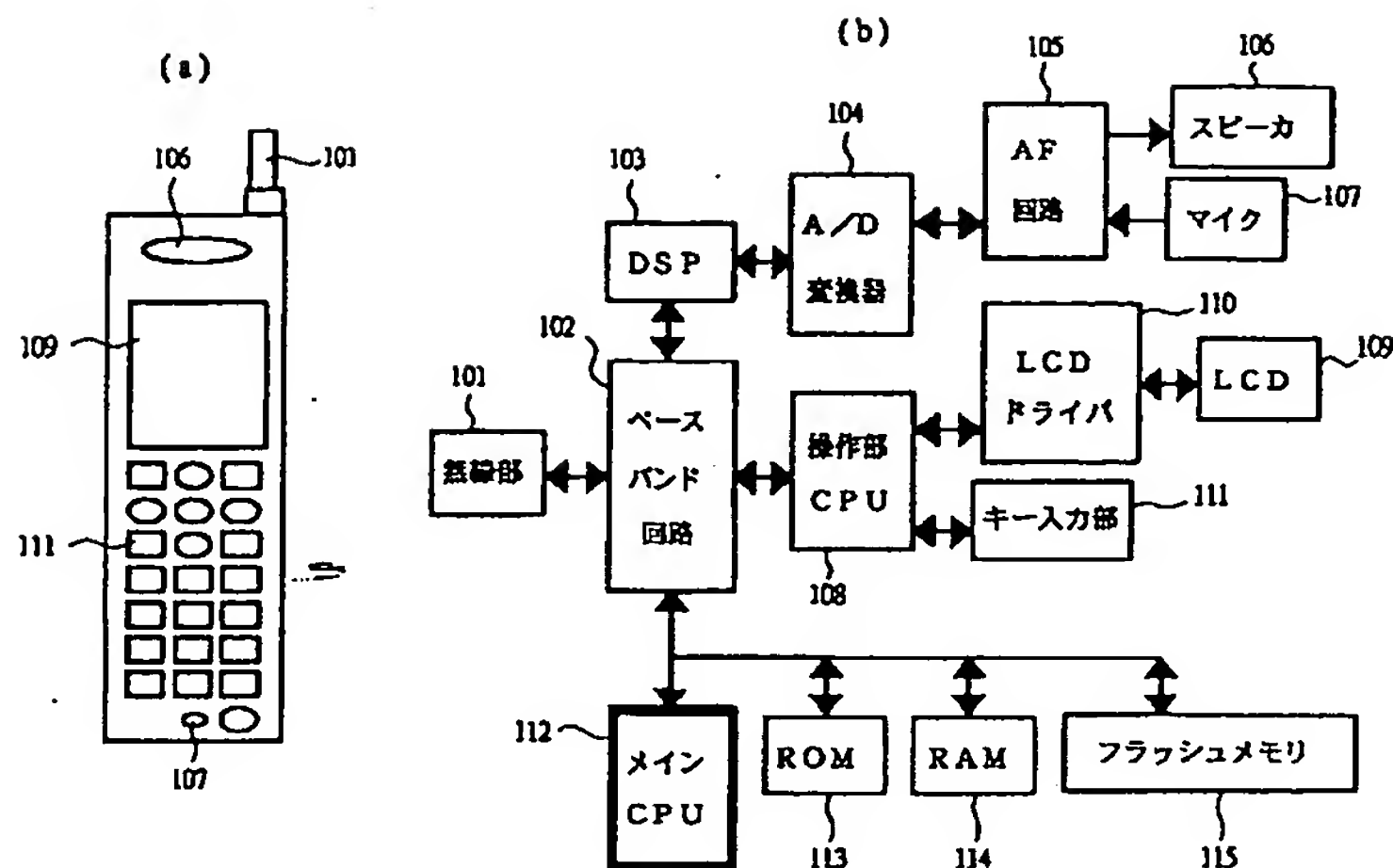
【符号の説明】

- 1 メインCPU
- 2 主記憶装置
- 3 外部バス
- 4 ROM
- 5 RAM
- 6 スタックポインタ
- 10 データキャッシュ
- 11 命令キャッシュ
- 12 ライトバッファ
- 13 制御ユニット

- 14 命令バッファ
- 15 命令デコードユニット
- 16 レジスタファイル
- 17 プログラムカウンタ
- 18 演算回路
- 19 メモリデータアクセスユニット
- 20 ライトバックユニット
- 21 バスユニット
- 22 内部アドレスバス
- 23 内部データバス
- 24 外部アドレスバス
- 25 外部データバス
- 26 外部制御信号線
- 40 ADDRESS
- 41 TAG
- 42 INDEX
- 43 OFFSET
- 44 インデックスデコーダ
- 45 Tag
- 46 Validビット
- 47 Dirtyビット
- 48 データブロック
- 49 LRUビット
- 50 キャッシュ制御ユニット
- 60 ADDRESS
- 61 TAG
- 62 INDEX
- 63 OFFSET
- 64 Way J
- 101 無線部
- 102 ベースバンド回路
- 103 DSP
- 104 A/D変換器
- 105 AF回路
- 106 スピーカ
- 107 マイク
- 108 操作部・CPU
- 109 LCD
- 110 LCDドライバ
- 111 キー入力部
- 112 メインCPU
- 113 ROM
- 114 RAM
- 115 フラッシュメモリ

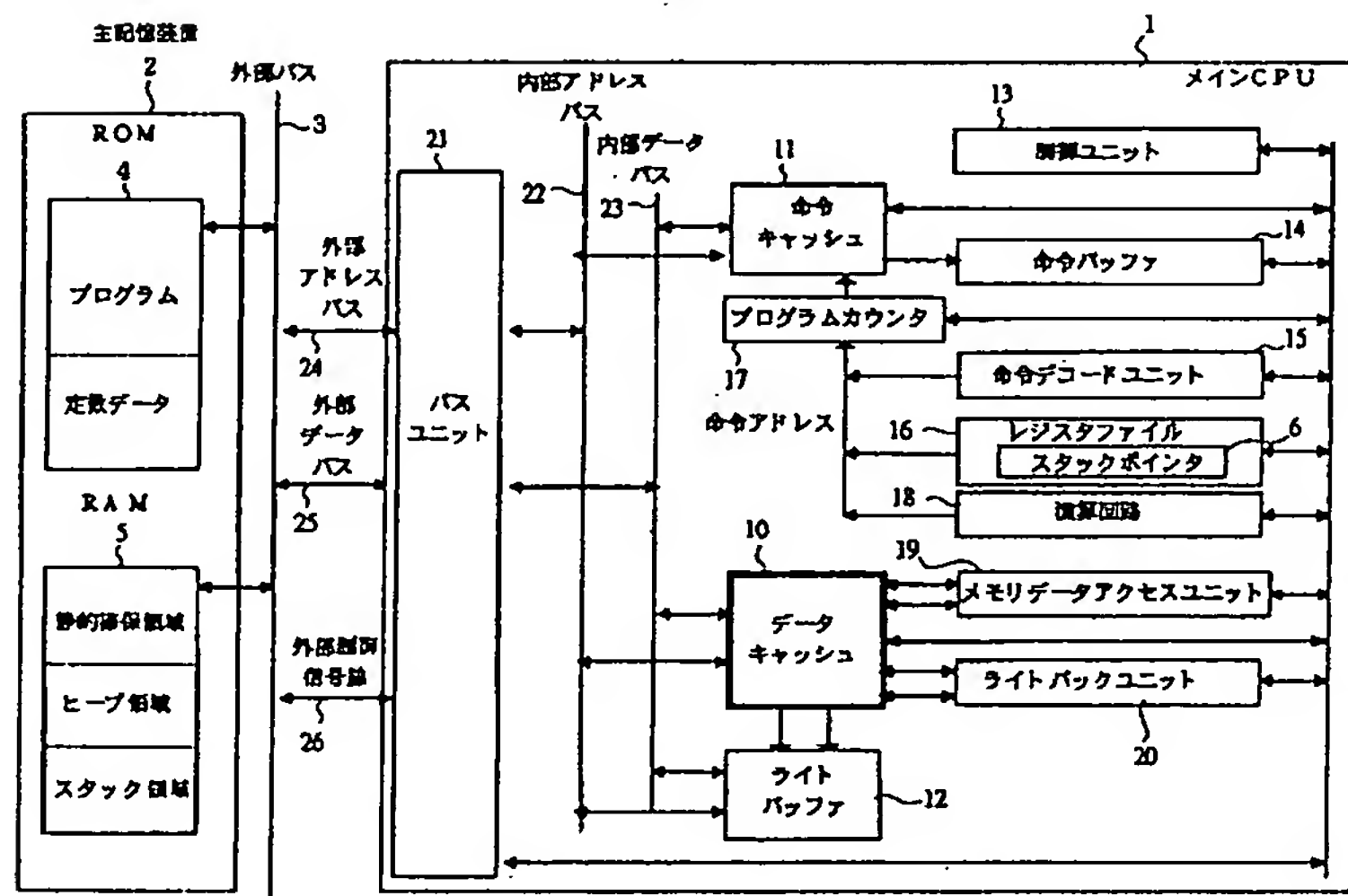
【図1】

図 1



【図2】

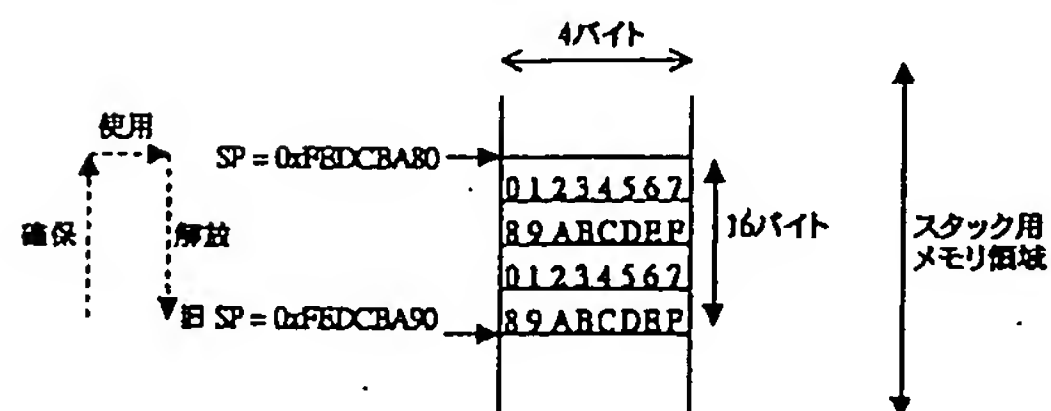
図 2



【図10】

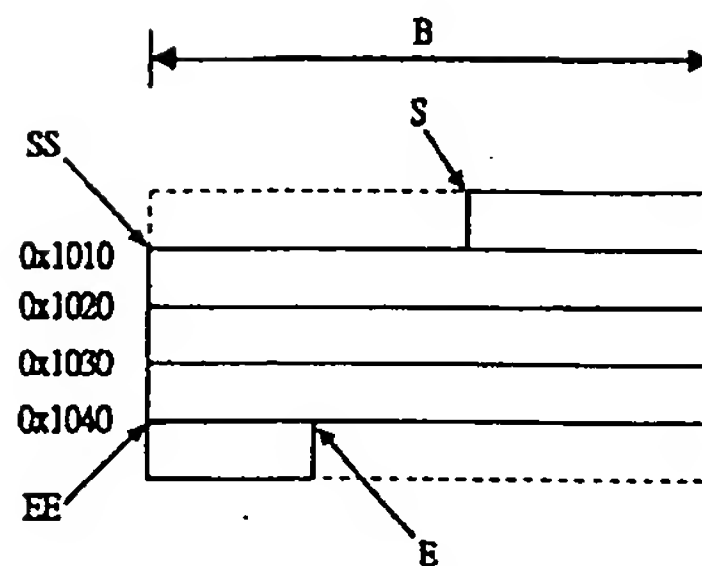
図 10

主記憶装置のRAM



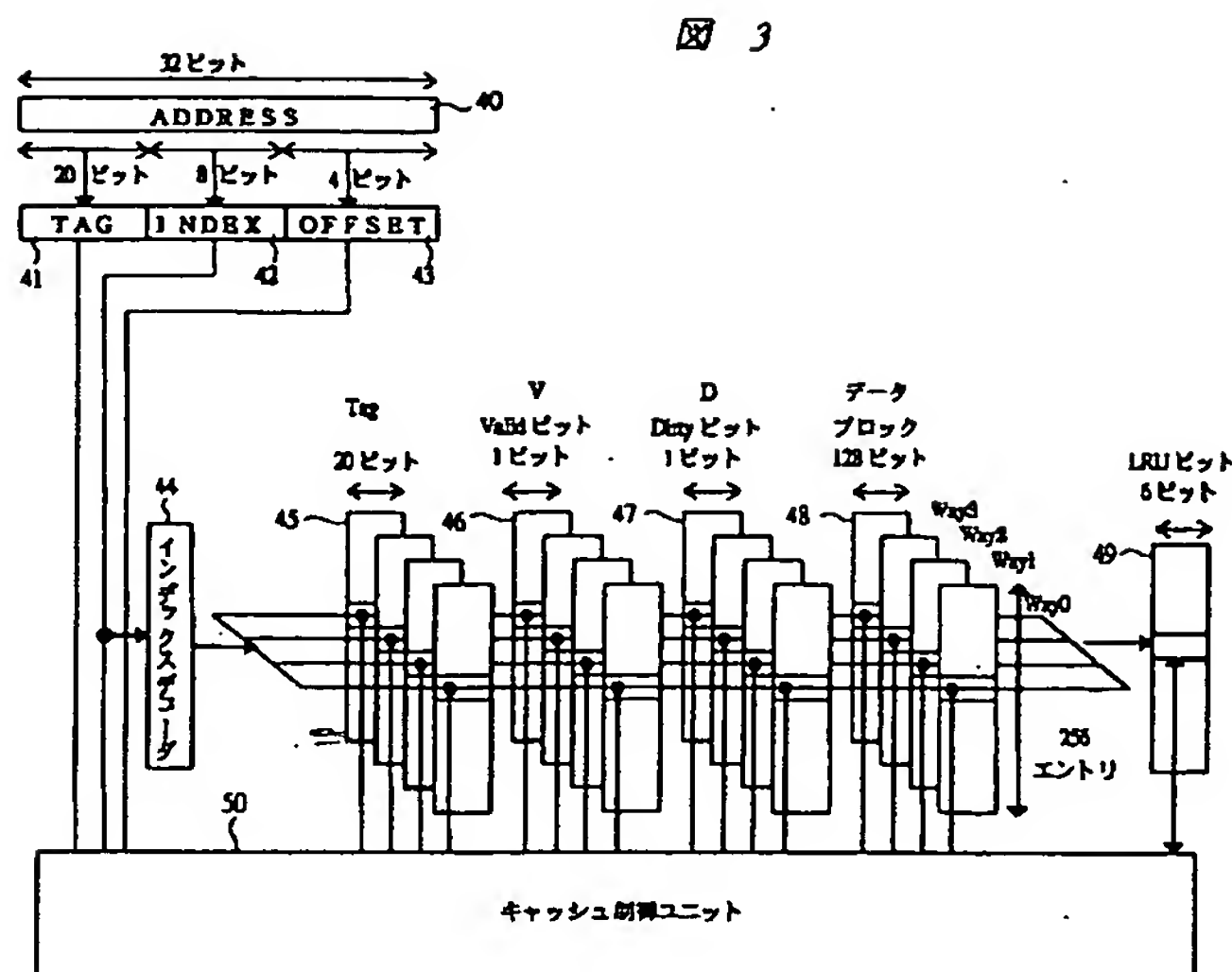
【図7】

図 7

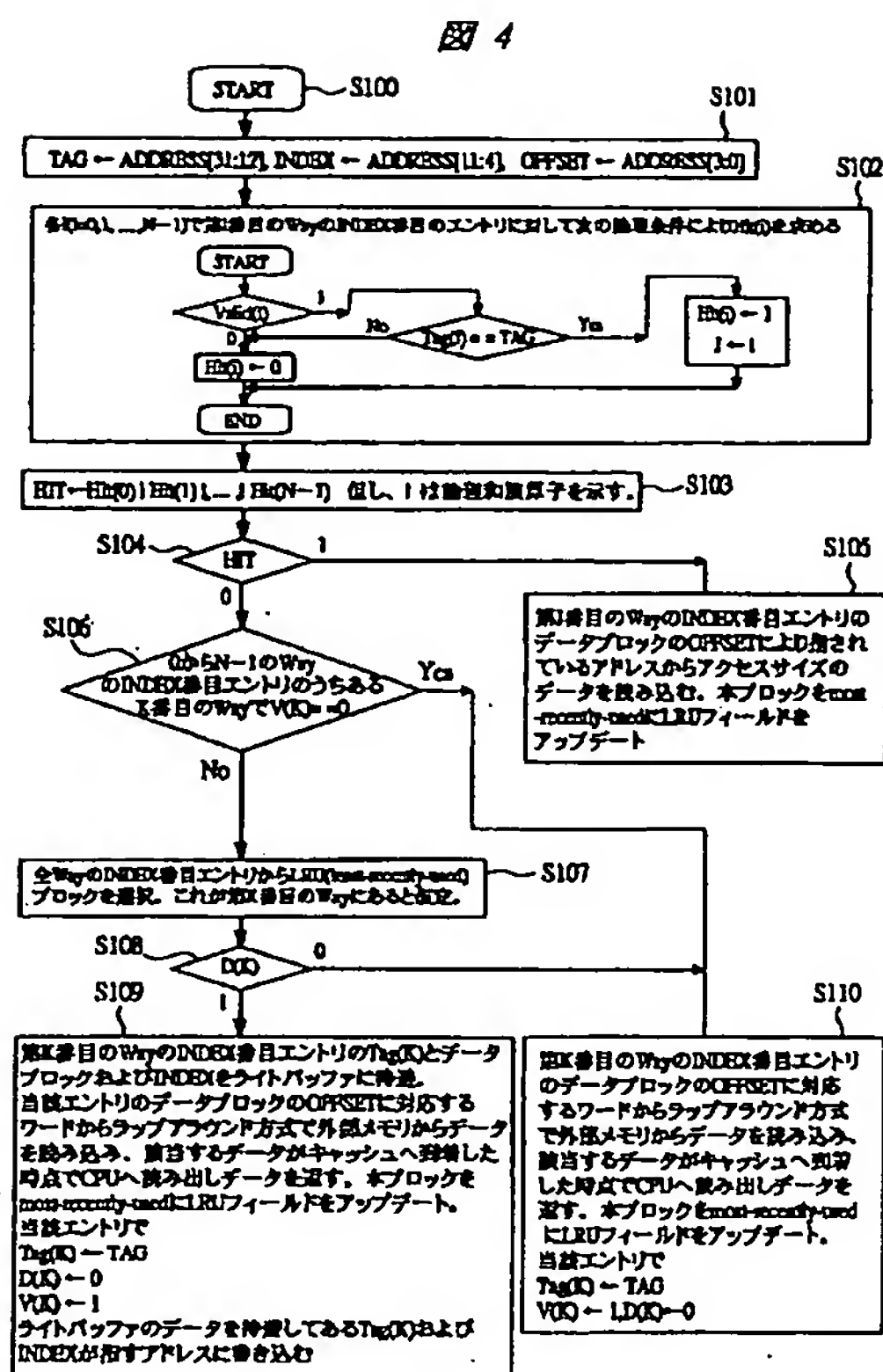




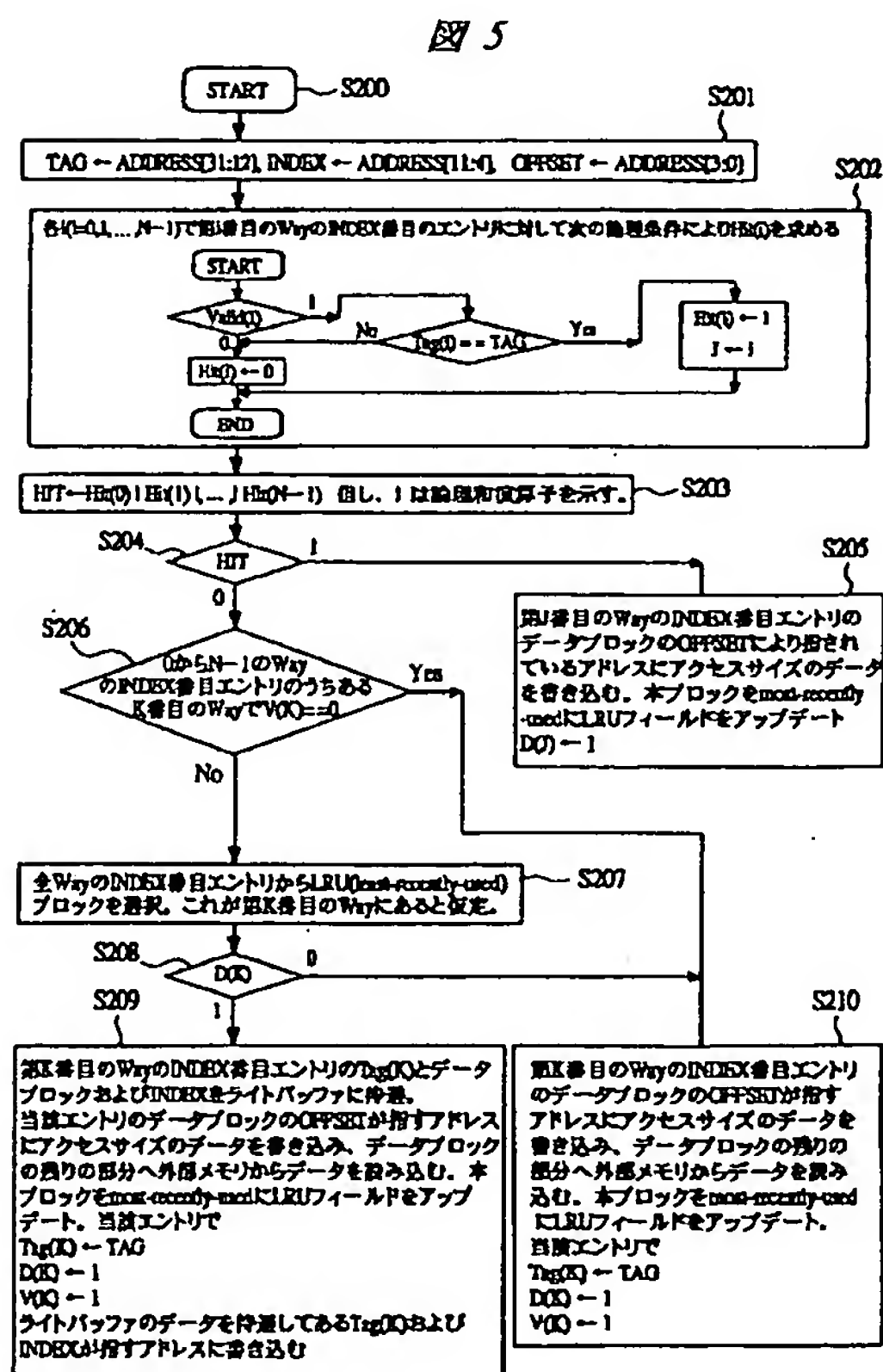
【図3】



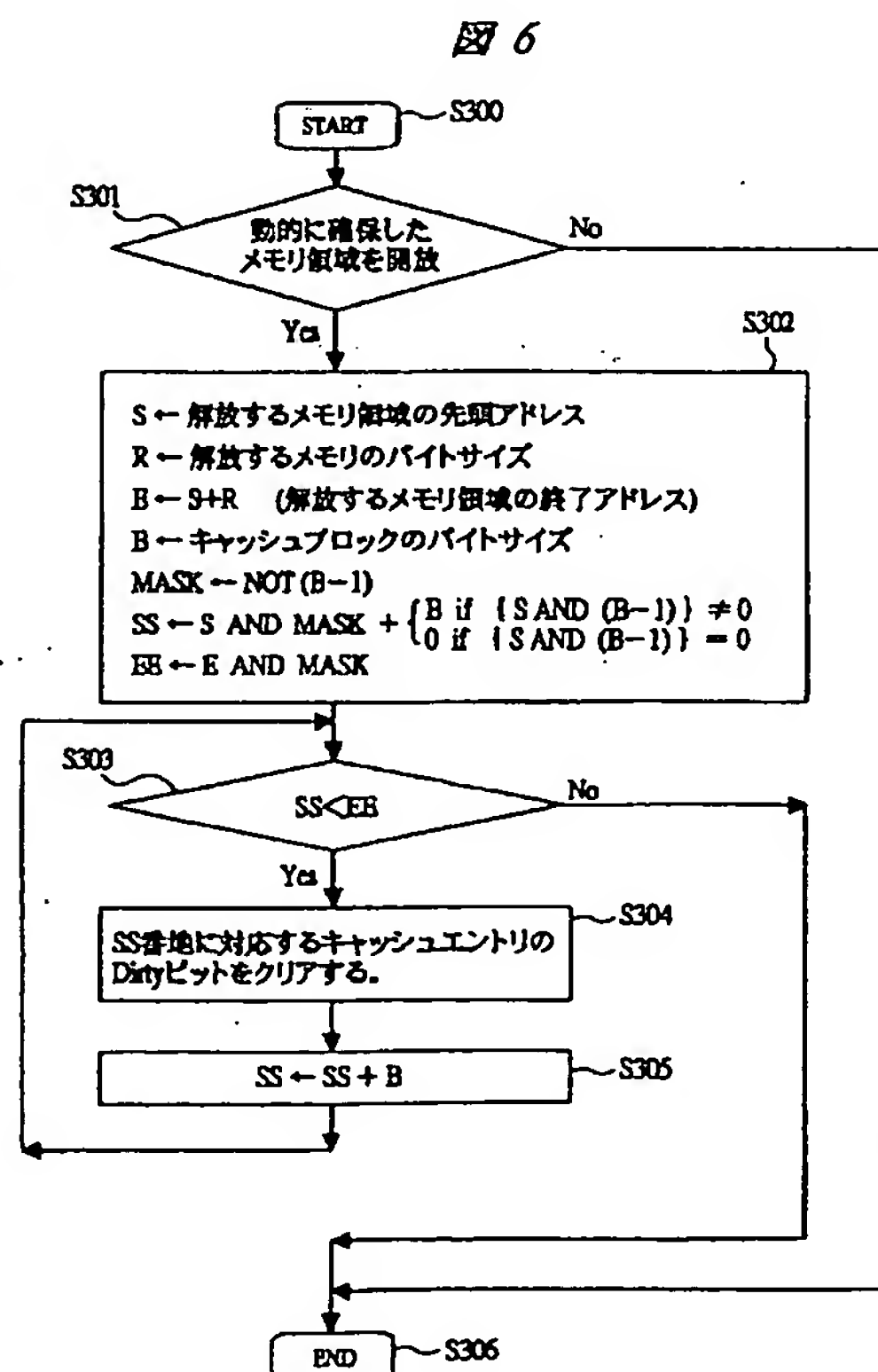
【図4】



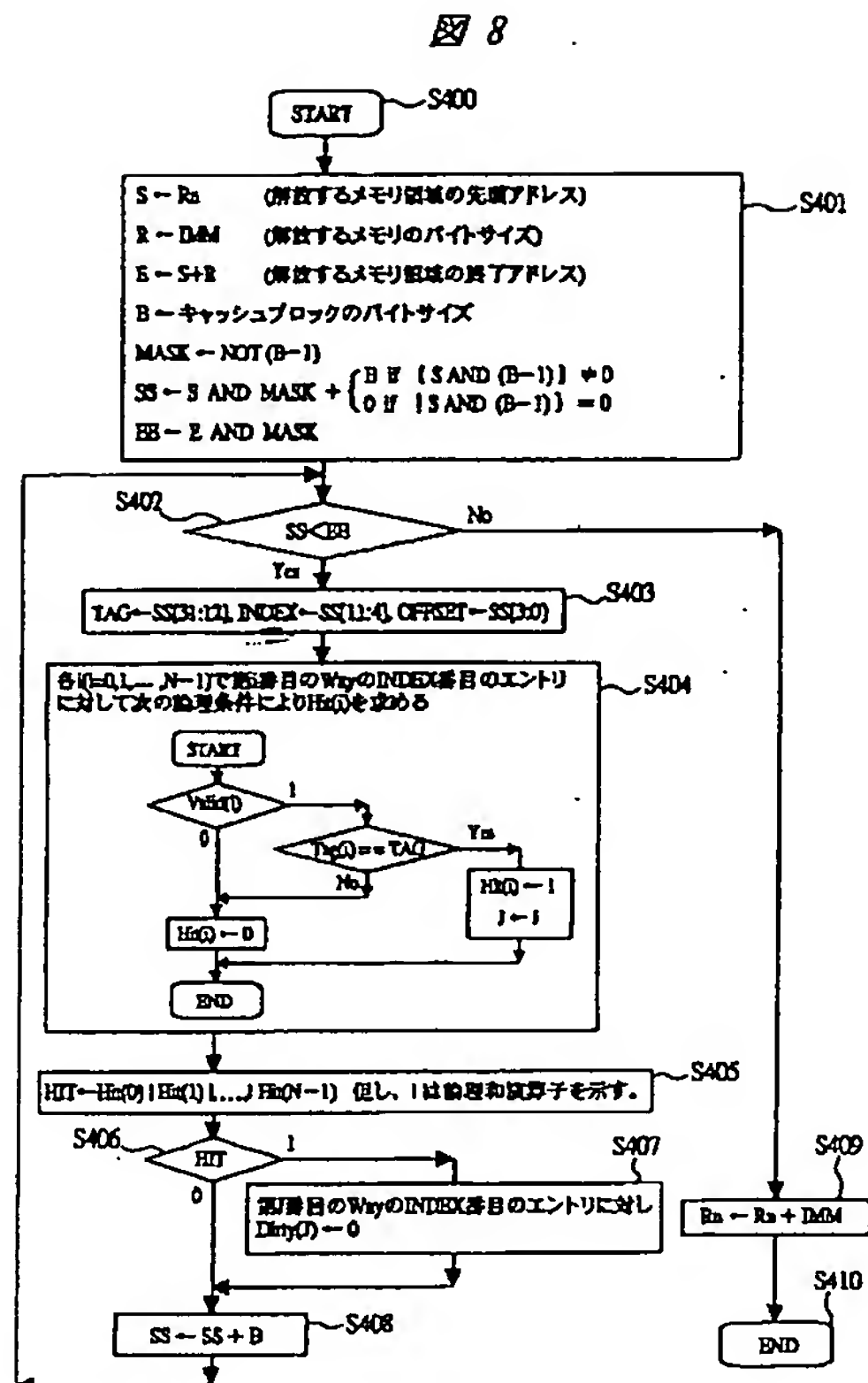
【図5】



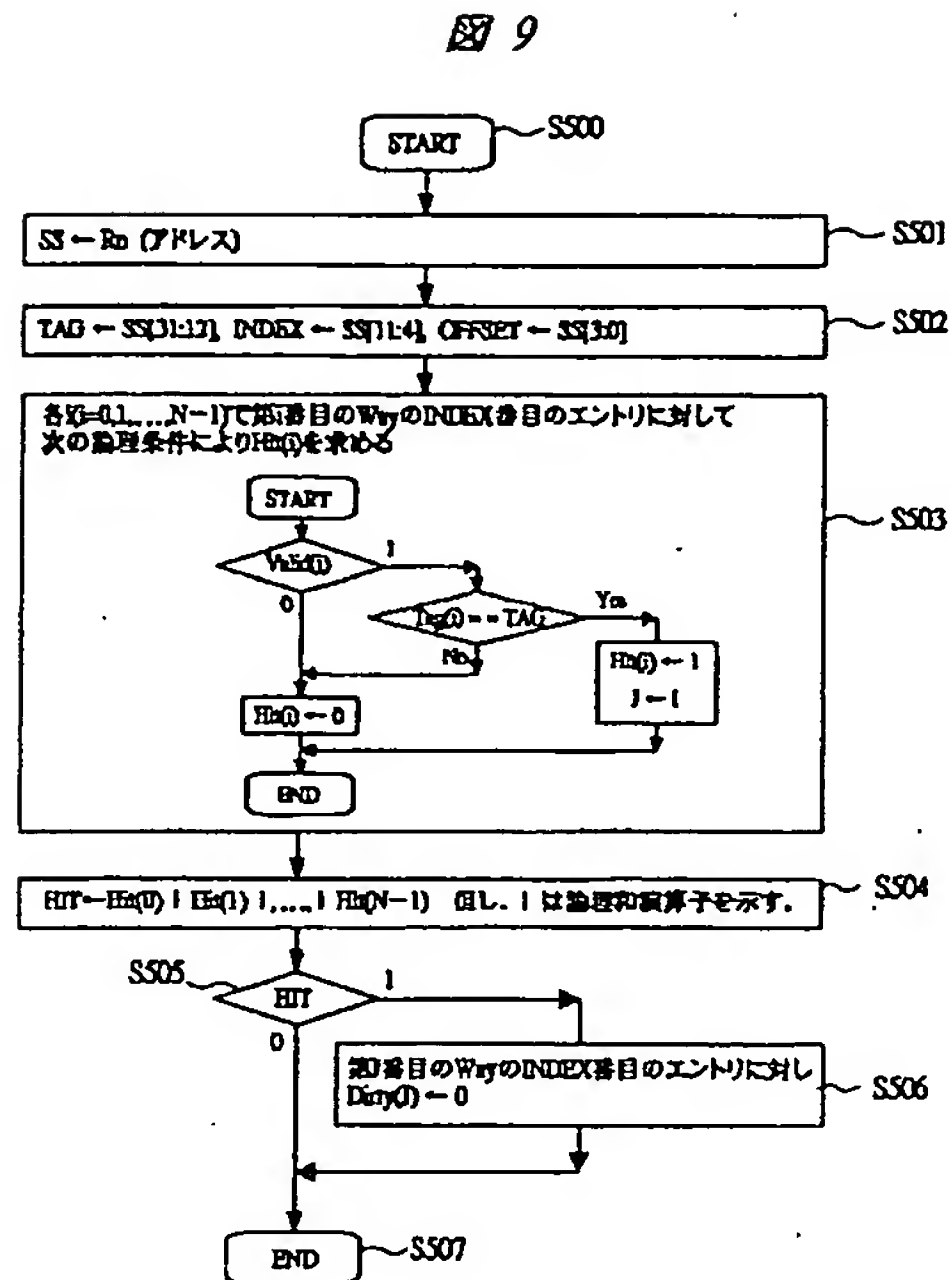
【図6】



【図8】

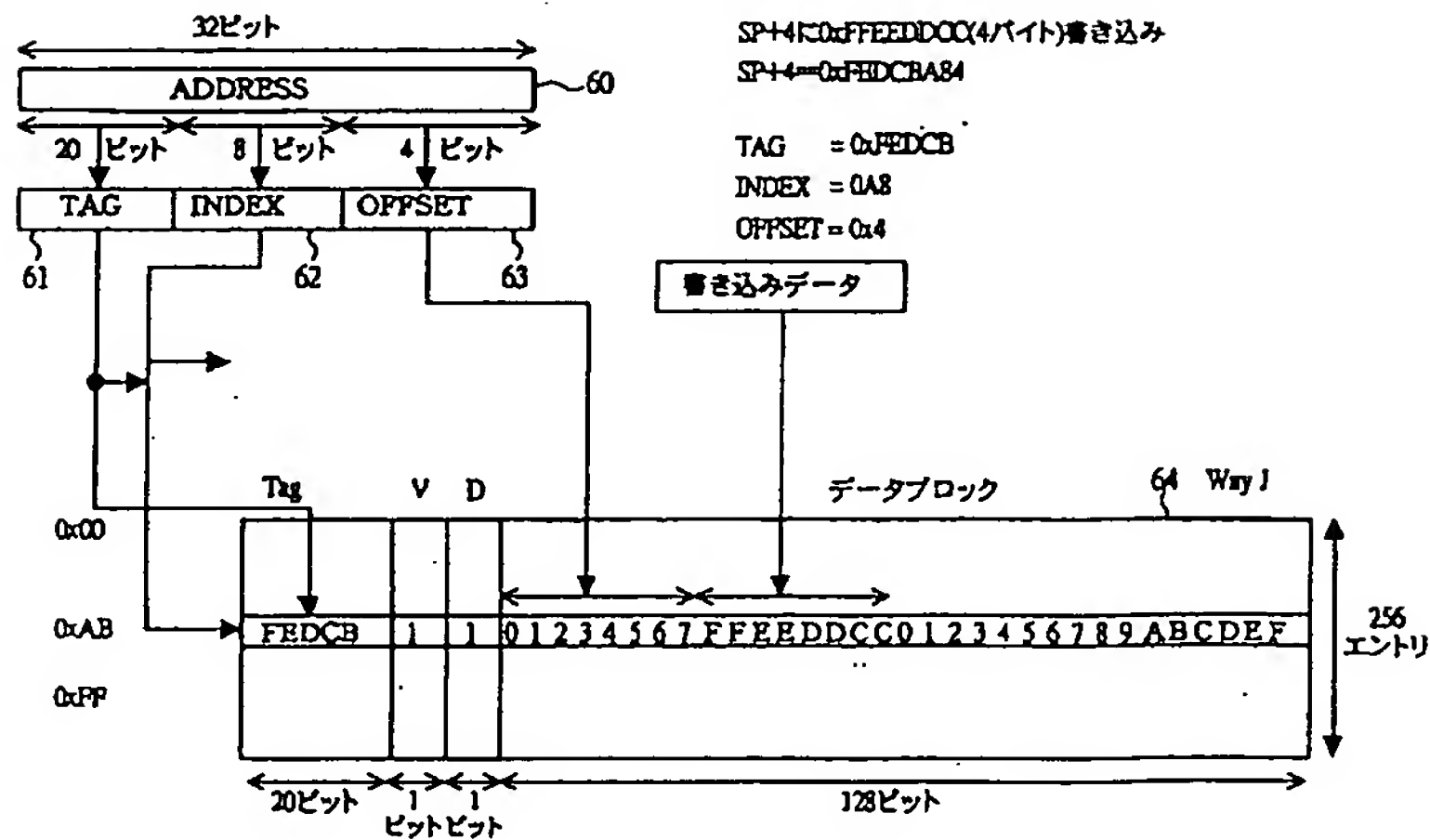


【図9】

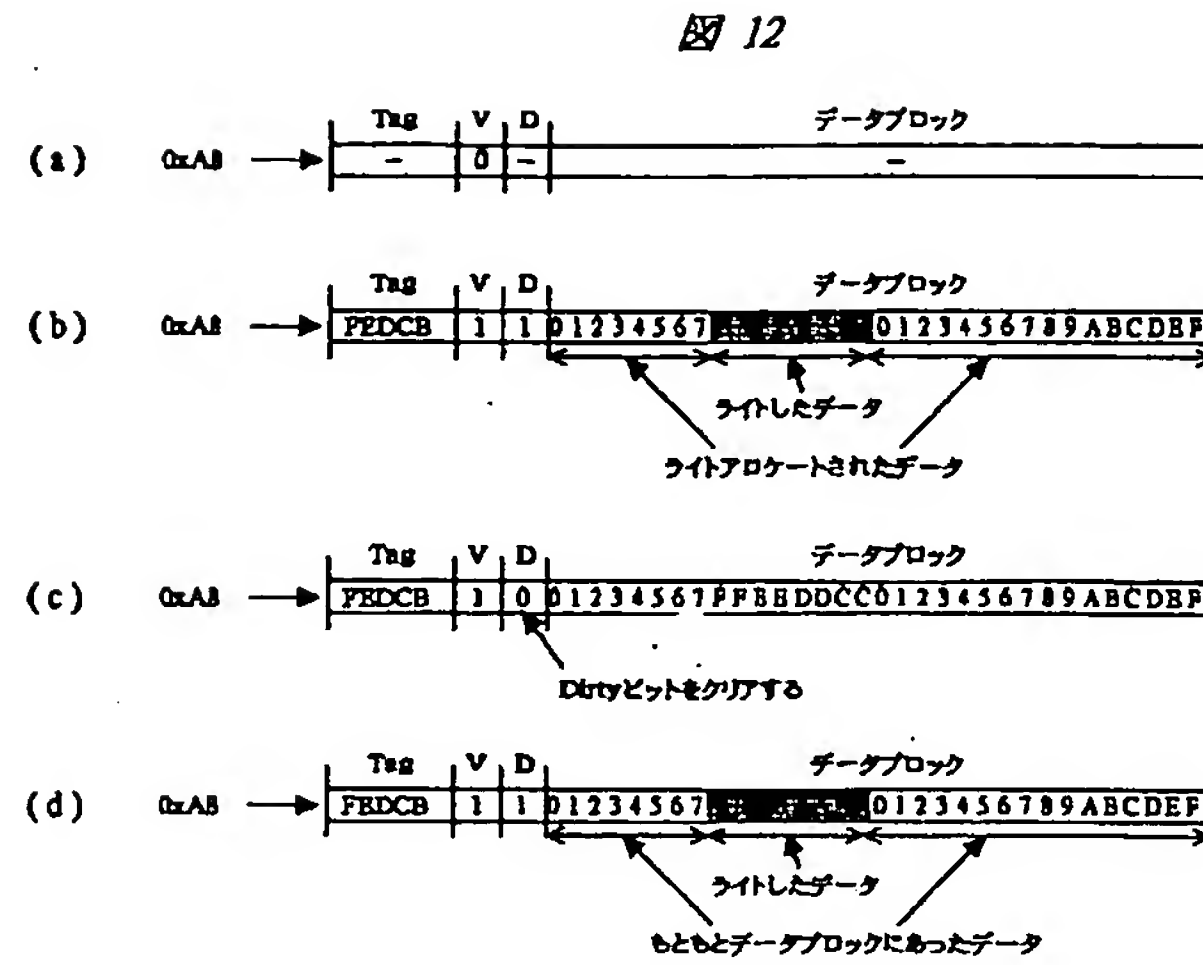


【図11】

図 11



【図12】



フロントページの続き

(51)Int.Cl. <sup>7</sup>		識別記号		F I		サーチコード (参考)
G 0 6 F	12/08	5 7 9		G 0 6 F	12/08	5 7 9
	12/12	5 0 3			12/12	5 0 3



**Japanese Laid-Open Patent Application No. 2003-223360**

[What is claimed is]

[Claim 1] A cache memory system comprising  
a write back method cache memory,  
wherein said cache memory system can execute a command for manipulating a  
dirty bit of a reserved area in said cache memory.

[Claim 2] The cache memory system according to Claim 1,  
wherein the command is a memory releasing command, and  
a mnemonic and operand of the memory release command are:  
MREL Rn, IMM  
Rn: Register name  
IMM: immediate value having a predetermined number of bits.

[Claim 3] The cache memory system according to Claim 1,  
wherein the command is a dirty bit clearing command, and  
the mnemonic and operand of the dirty bit clearing command are:  
DCBDC @Rn  
Rn: Register name.

[Claim 4] The cache memory system according to one of Claim 2 and Claim 3,  
wherein, the Rn: Register name is a register that can be used for releasing a  
dynamically reserved memory area in a stack area.

[Claim 5] The cache memory system according to one of Claim 2 and Claim 3,  
wherein, the Rn: Register name is a register that can be used for releasing a  
dynamically reserved memory area in a heap area.

[Claim 6] The cache memory system according to Claim 1, further comprising  
a processor operable to instruct execution of the command; and  
a main storage device,  
wherein a region identified by a specific address in said main storage device is  
reserved and a predetermined program process is performed, based on an instruction  
from said processor, and



when, after the reserved area is reserved in said cache memory and used, the reserved area in said main storage device is to be cleared according to an instruction from said processor, control is performed so as to clear a dirty bit of the region reserved in said cache memory.

[Claim 7] A microprocessor comprising:

a write back method cache memory,

wherein said cache memory system can execute a command for manipulating a dirty bit of a reserved area in said cache memory.

[0001]

[Field of the Invention]

The present invention relates to cache memory control technology, and particularly to effective technology applied to a cache memory control method for releasing a dynamically reserved memory area in a cache memory system that has a cache memory, and the like, of a micro processor, a micro computer, or a micro controller.

[0002]

[Description of the Related Art]

According to the consideration of the inventor, cache memory control technology is cited in technology described in the respective publications of Japanese Laid-Open Patent Application No.11-338772, Japanese Laid-Open Patent Application No.4-268638, Japanese Laid-Open Patent Application No.4-264641, Japanese Laid-Open Patent Application No.4-188326, Japanese Laid-Open Patent Application No.6-28253, and Japanese Laid-Open Patent Application No.9-231133.

[0003]

In the above-mentioned technology included in the publications, when in the condition in which an area reserved dynamically in the main memory is read as a cache memory entry, such area of the main memory is released, such cache entry is made invalid by clearing a valid bit of the cache entry.

[0004]

[Problems that the Invention is to Solve]

As a result that the inventor has considered a cache memory control technology



such as the above-mentioned technology, the following has become apparent. The following describes the technology that the inventor has examined as the prerequisites of the present invention with reference to Figure 10 to 12. Figure 10 is an explanatory diagram of the stack usage, Figure 11 the cache condition, and Figure 12 the cache entry with an INDEX as 0xA8.

[0005]

[Description of General Stack Usage: Figure 10]

Assume that the stack pointer SP before using has the value of  $SP=0xFEDCBA90$ . 0x is a prefix which indicates that the successive character string is hexadecimal. When reserving a 0x10-byte area in the stack,  $SP \leftarrow SP-0x10$  is performed, whereby  $SP=0xFEDCBA80$  is obtained.

[0006]

The 0x10-byte memory area between User Old  $SP=0xFEDCBA90$  and Current  $SP=0xFEDCBA80$  is available. For example,

Data 0x01234567 is written to the address (SP+0);

Data 0x89ABCDEF is written to the address (SP+4);

Data 0x01234567 is written to the address (SP+8);

Data 0x89ABCDEF is written to the address (SP+12).

When these data become unnecessary and a memory area for these data no longer needs to be reserved,  $SP \leftarrow SP+0x10$  is performed. Then, SP returns to the value of Old SP and the reserved 0x10-byte memory area is released.

[0007]

Memory areas in addresses lower than the SP in stack memory areas are unused and released. Among stack memory areas, memory areas that have higher addresses than the SP are already reserved.

[0008]

[Cache operation during writing: Figure 11]

Figure 11 shows the cache condition after data 0xFFEEDDCC is written into the address (SP+4), that is, into the address 0xFEDCBA84.

[0009]

The address 0xFEDCBA84 to be written is treated as ADDRESS60, and the 20 bits

from the MSB of the ADDRESS is treated as TAG 61, the next 8 bits as INDEX 62 and the last 4 bits as OFFSET 63. It is assumed that the entry of INDEX=0 x A8 of Way J 64 is in the Invalid condition (Valid bit V=0) before being written and the Way J was selected for this writing operation. In this entry, when V=0, although undefined values are included in Tag, Dirty bit (D bit) and a data block, 0 x FEDCB is written into Tag, 1 in the V bit, 1 in the D bit, and data 0xFFEEDDCC in 4 bytes from the position that the data block offset is OFFSET=4.

[0010]

Since this description assumes a cache memory based on the write-back method, when there is no coherence between the main memory and the cache entry's data block, D bit is set to 1. Data from the corresponding addresses of the main memory can be read in the rest of this data block by using the write allocate feature. Figure 11 shows this condition.

[0011]

[Specific Description of Prerequisite Technology: Figure 12]

In the prerequisite technology of the present invention,  $SP \leftarrow SP + 0x10$  releases the stack memory area of 0x10 bytes and at the same time the valid bit (V bit) of the cache entry allocated with a released memory area is cleared to 0 to set Invalid. In the case that V bit is not cleared, when this cache entry is evicted from the cache memory according to the LRU algorithm, if D bit is 1, the content of the data block is written back to the main memory. However, because the content of this data block is basically the data of an area that has been already released, it is not reused even if it is written back to the main memory. So such write back is useless. Therefore, useless write back has been prevented by clearing V bit in order to avoid write back. Figure 12 (a) shows the Invalid condition of the cache entry.

[0012]

[Problems of the Prerequisite Technology of the Invention]

The problem of the above-mentioned prerequisite technology of the present invention is revealed when the address, which is the same as that of an area that was released once, is reserved again in the write-allocate method cache memory. In the prerequisite technology of the present invention, because the Valid bit is cleared when the area is released, the cache entry becomes invalid and the tag and data are lost. When the same address range is reserved again and is written again, new allocation of a cache entry is

necessary because the tag has already been lost. According to the write-allocate method, the cache entry data, except the written data, is read into the cache entry from the main memory. This data transfer may cause power consumption and the congestion of data routes, leading to performance deterioration. Solving these problems is the object of the present invention.

[0013]

Assuming that when a 16-byte stack area beginning with 0xFEDCBA80 is released, the cache entry is set to Invalid as shown in Figure 12 (a) based on the prerequisite technology of the present invention. Next, when it is assumed that the value of 0BBBBBAAAA is written into the address of: 0xFEDCBA84, which is allocated to the same Way, the result is shown in Figure 12 (b). In this case, the part other than the 4-byte written data of the data block is read from the main memory. The object of the present invention is to reduce power consumption and eliminate the congestion of data transfer routes caused by this reading operation, and thus improving throughput.

[0014]

In other words, the object of the present invention is to reduce unnecessary data transfer between the main memory and the cache memory as well as power consumption along with data transfer, and alleviate the congestion of data transfer routes, improving throughput.

[0015]

The above object of the present invention as well as other objects and new features shall become clear with reference to the description specification and attached figures.

[0016]

[Means to Solve the Problems]

The overview of some typical inventions to be disclosed in the present invention is briefly described hereinafter.

[0017]

The cache memory system according to the present invention is a cache memory system including a write back cache memory, wherein said cache memory system can execute a command for manipulating a dirty bit in said cache memory. In particular, the command is a memory releasing command MREL Rn, IMM, or a dirty bit clearing



command DCBDC @Rn, which is applied in a register which can be used for releasing a dynamically reserved memory area in a stack area or a heap area.

[0018]

In addition, in the above-mentioned cache memory system, a processor directing the implementation of an order and the main memory are included. Based on the direction of the processor, an area indicated by the specific address within the main memory is reserved and the predetermined program is executed. After this reserved area is reserved on the cache memory and is used, when the reserved area in the main memory is released according to the direction of the processor, the Dirty bit of the area reserved on the cache memory is controlled so as to be cleared.

[0019]

In other words, while, in the prerequisite technology of the present invention, the cache entry Valid bit corresponding to the released area of the main memory is cleared, the present invention is characterized in that the cache entry Dirty bit is cleared.

[0020]

Consequently, in the case where the cache entry in which the Dirty bit is cleared is present in the cache, a main memory area in the same address range is newly reserved and, when data is first written to this area, the effects of the present invention are seen. As the cache entry corresponding to the address range is already present in the cache, there is no need for allocating data to be written and no data transfer takes place. Therefore, there is no power consumption caused by data transfer and no performance deterioration caused by the congestion of data transfer routes.

[0021]

In addition, in the case that the cache entry in which the Dirty bit is cleared is evicted from the cache, because the Dirty bit is cleared, the cache entry data need not be written back to the main memory and no data transfer takes place for the write back. Therefore, there is no power consumption caused by data transfer and no performance deterioration caused by the congestion of data transfer routes.

[0022]

For example, in the condition where a 16-byte stack area beginning with 0xFEDCBA80 is allocated to the cache entry of INDEX=0xA8 of Way J as shown in the

above-mentioned Figure 11, when the area is released, in the present invention, the cache entry Dirty bit is set to 0 as shown in Figure 12 (c). If the value of 0xBBBAAAA is written to the address of 0xFEDCBA84 before this cache entry is evicted from the cache, the condition of Figure 12 (d) is provided. The part other than the written data of the data block is never read from the main memory and the part is the value originally written in the data block. Therefore, although in the prerequisite technology of the present invention, reading from the main memory takes place based on the write-allocate method, this reading does not take place in this method. Power consumption caused by this reading is reduced and the congestion of data transfer routes caused by this read-in does not take place, resulting in the improvement of throughput.

[0023]

In addition, when the cache entry in the condition of Figure 12 (c) is evicted from the cache based on the LRU algorithm, because the Dirty bit is cleared to 0, there is no power consumption caused by data transfer for write back and no performance deterioration caused by the congestion of data transfer routes.

[0067]

[First Embodiment]

The embodiment is described with reference to Figure 6, 7 and 8. Figure 6 is a flow chart indicating the cache control method of the dynamically reserved area; Figure 7 describes variables used in Figure 6 and 7; Figure 8 is a flow chart indicating the order behavior for releasing a memory.

[0068]

In the present embodiment, a CPU in which the above-mentioned data cache is implemented is discussed. The write-back method and the write-allocate method are assumed to be used.

[0069]

The description of the embodiment is for realizing the algorithm of the cache control method in the dynamically reserved area as shown in Figure 6. In the example of the description of the embodiment, the judgment in step S 301 of Figure 6 is assumed to have already been performed by a compiler or a programmer. In addition, as shown in step S 301 of Figure 6, a function is added to add the register value enough to make up for the released value in order to use it for the release of the stack.

[0070]

In other words, in Figure 6, soon after the start (step S 300), the cache control method of a dynamically reserved area judges if a dynamically reserved memory area is released or not in the judgment process to judge the dynamically reserved area in step S 301. When the memory area is not released (No), the process is terminated (step S 306) and when the memory area is released (Yes), the process moves on to step S 302 and the successive steps.

[0071]

In the initialization process of step S 302, the following assignment is done for variables including S, R, E, B, MASK, SS, and EE: the initial address of a memory area to be released  $\rightarrow$  S; the size of a memory area to be released  $\rightarrow$  R;  $S + R$  (the ending address of a memory area to be released)  $\rightarrow$  E; the byte size of a cache block  $\rightarrow$  B;  $\text{NOT}(B-1) \rightarrow \text{MASK}$ ;  $S \text{ AND } \text{MASK} + \{B \text{ if } \{S \text{ AND } (B-1)\} \neq 0, 0 \text{ if } \{S \text{ AND } (B-1)\} = 0\} \rightarrow \text{SS}$ ;  $E \text{ AND } \text{MASK} \rightarrow \text{EE}$ . NOT x represents an inversion per bit of x at a 32 bit width. x AND y represents the conjunction per bit of x and y at a 32 bit width.

[0072]

In addition, in the discrimination process of the loop termination of step S 303, if the discrimination of  $\text{SS} < \text{EE}$  is NO, the process is terminated. If the discrimination is YES, a cache entry Dirty bit corresponding to the address SS is cleared in the Dirty bit clear process. Subsequently, after  $\text{SS} + B$  is assigned to SS in the address counter update process of step S 305, step S 303 and the successive processes are repeated.

[0073]

Figure 7 describes the above-mentioned variables such as S, R, E, B, MASK, SS, and EE.

[0074]

In Figure 7, for example, if the variable corresponding to the initial address of a memory area to be released is  $S = 0x100C$ , the variable corresponding to the byte size of a memory to be released is  $R = 0x38$ , the variable corresponding to the ending address of a memory area to be released is  $E = S + R = 0x100C + 0x38 = 0x1044$ , and the variable corresponding to the byte size of a cache block is  $B = 16$ , the correction variable  $\text{MASK} = \text{NOT}(B-1) = \text{NOT}(0xF) = 0xFFFFFFFF0$ , and the variable after correction  $\text{SS} = 0x1010$  and

the variable EE=0x1040 are provided.

[0075]

In other words, SS is the number that S (the initial address of a memory area to be released =0x100C) is rounded out to the multiple number of B (the cache block byte size =16), and EE is the number that E (the ending address of a memory area to be released =0x1040) is truncated to the multiple number of B. Namely, in Figure 7, because a dashed lined memory area may be used for other processes, it needs to be excluded.

[0076]

In the description of the embodiment, the present invention is applied as the instruction for supporting the release of a stack area. The mnemonic and operand of this memory release order are shown as follows.

[0077]

MREL Rn, IMM

Rn: register name

IMM: the predetermined immediate bit number

MREL order implements the behavior represented in the behavior algorithm of the memory release order MREL, Rn, and IMM of Figure 8.

[0078]

For example, in the case of releasing a stack area immediately before returning it from a function, when the stack pointer is R15 and a memory size to be released is 32 bytes, according to MREL R15, 32,  $R15 \leftarrow R15 + 32$  is performed, releasing a 32 byte area of the stack. At the same time, the Dirty bit of a cache entry area between the pre-change R15 and the post-change R15 is cleared. When Dirty bit is cleared, as described in step S 401 and 402 of Figure 8, the release range is calculated, disregarding a cache entry including an address that is not the release target.

[0079]

In other words, in Figure 8, the behavior of a memory release order MREL Rn, IMM, is the implementation of the following assignment to variables including S, R, E, B, MASK, SS, and EE in the initialization process of step S 401 after the start (step S 400): the initial address of a memory area to be released,  $Rn \rightarrow S$ ; the byte size of a memory



area to be released,  $IMM \rightarrow R$ ; the ending address of a memory area to be released,  $(S+R) \rightarrow E$ ; the byte size of a cache block  $\rightarrow B$ ;  $NOT (B-1) \rightarrow MASK$ ;  $S AND MASK + \{B \text{ if } \{S AND (B-1)\} \neq 0, 0 \text{ if } \{S AND (B-1)\} = 0\} \rightarrow SS$ ;  $E AND MASK \rightarrow EE$ .

[0080]

In addition, in the discrimination process of the loop termination of step S 402, if the discrimination of  $SS < EE$  is No, the assignment of  $R_n + IMM \rightarrow R_n$  is performed in the register update process of step S 409 and the process is terminated (step S 410). If the discrimination is Yes, the process moves on to step S 403 and the successive processes.

[0081]

In the address decomposition process of step S 403,  $SS [31:12]$ ,  $SS [11:4]$  and  $SS[3:0]$  are inputted to TAG, INDEX and OFFSET respectively. In addition, in the cache hit discrimination process of step S 404, Hit (i) is calculated according to the logic requirement that when  $Valid (i)=1$  and  $Tag = TAG_i \rightarrow Hit (i)$ ,  $i \rightarrow J$ ,  $Valid (i)=0$ , or  $Valid (i)=1$  but not  $Tag = TAG_i \rightarrow Hit (i)$  is provided for the  $INDEX^{th}$  entry of the  $i^{th}$  Way at each i ( $i = 0, 1, \dots, N - 1$ ). Then, in the cache hit calculation process of step S 405, the logical OR of the calculated Hit (0), Hit (1), ..., Hit (N-1) is calculated and assigned to IT.

[0082]

In addition, in the cache hit discrimination process of step S 406, after HIT is discriminated, when  $HIT = 1$ , namely there is a hit,  $0 \rightarrow Dirty (J)$  is assigned to the  $INDEX^{th}$  entry of the  $J^{th}$  Way in the Dirty bit clear process of step S 407. Then the process moves on to step S 408.

[0083]

In addition, when  $HIT=0$ , namely there is no hit, after the assignment of  $SS+B \rightarrow SS$  is performed in the address counter update process, step S 402 and the successive processes are repeatedly performed.

[0084]

In addition, in the case of assigning a stack area exclusively allocated to each task based on the multitask method, the examples of the description of the embodiment can be applied without modification.

[0085]

[Second Embodiment]

The description of the embodiment is a deformation example of the above-mentioned First Embodiment. As shown in the above-mentioned First Embodiment, a register to be specified to the MREL order is not limited to a register used for a stack pointer. It can be used when a dynamically reserved memory area in a heap area is released.

[0086]

Specifically, there is an example that the MREL order is implemented for an area released inside the free function in the standard library functions of the C language. If the initial address of an area to be released is R1 and the byte number to be released is 100, the Dirty bit hit of a cache entry corresponding to the released area is cleared by implementing MREL R1, 100.

[0087]

[Third Embodiment]

The description of the embodiment is shown with reference to Figure 9. Figure 9 is a flow chart indicating the behavior of the Dirty bit clear order.

[0088]

In the description of the embodiment, step S 304 of the above-mentioned Figure 6 is realized as the order of a CPU. The mnemonic and operand of the Dirty bit clear order are shown as follows.

[0089]

DCBDC @Rn

Rn: register name

The DCBDC order implements the behavior represented in the behavior algorithm of DCBDC @Rn, the Dirty bit clear order of Figure 9

[0090]

In other words, in Figure 9, after the start (step S 500), the behavior of the Dirty bit clear order DCBDC @Rn assigns Rn (address) to SS in the process of obtaining the target address of step S 501.

[0091]

In addition, in the address decomposition process of step S 502, SS [31:12], SS [11:4] and SS[3:0] are assigned to TAG, INDEX and OFFSET respectively. Moreover, in the

cache hit discrimination process of step S 503, Hit (i) is calculated according to the logic requirement that when Valid (i)=1 and Tag == TAG,  $1 \rightarrow \text{Hit (i)}$ ,  $i \rightarrow J$ , Valid (i)=0, or Valid (i)=1 but not Tag == TAG,  $0 \rightarrow \text{Hit (i)}$  is provided for the INDEX<sup>th</sup> entry of the i<sup>th</sup> Way at each i (i = 0, 1, ..., N-1). Then, in the cache hit calculation process of step S 504, the logical OR of the calculated Hit (0), Hit (1), ..., Hit (N-1) is calculated and assigned to IT.

[0092]

In addition, in the cache hit discrimination process of step S 505, after HIT is discriminated, when HIT = 1, namely there is a hit,  $0 \rightarrow \text{Dirty (J)}$  is assigned to the INDEX<sup>th</sup> entry of the J<sup>th</sup> Way in the Dirty bit clear process of step S 506 and the process is terminated (step S 507). When HIT = 0, namely there is no hit, the process is terminated.

[0093]

Therefore, according to the above-mentioned description of the embodiment, because a cache entry Dirty bit corresponding to the released area of the main memory is cleared, in the case that the cache entry in which this Dirty bit was cleared is present in the cache, the main memory area in the same address range is newly reserved. When data is first written to this area, because a cache entry corresponding to the address range is present in the cache, there is no need of allocating data to be written and no data transfer takes place. Thus, there is no power consumption caused by data transfer, and no performance deterioration caused by the congestion of the data transfer routes.

[0094]

In addition, in the case that the cache entry which cleared the Dirty bit is evicted from the cache based on The LRU algorithm, because the Dirty bit is cleared, the cache entry data is not needed to be written back to the main memory and there is no data transfer for writing back. Therefore, power for data transfer is not consumed and there is no performance deterioration caused by the congestion of data transfer routes.

[0095]

So far the present invention by the inventor has been described based on the description of the embodiment. However, the present invention is not limited to the above-mentioned description of the embodiment and it goes without saying that various changes are possible without departing from the scope of the present invention.

[0096]

[Effects of the present invention]

The effects of some typical inventions to be disclosed in the claim are described as follows:

[0097]

(1) In the case that when the dynamically reserved main memory area is released, the same area is reserved again by clearing the corresponding cache entry Dirty bit, write-allocate does not take place in the same cache entry, allowing the removal of data transfer between the cache memory and the main memory for write-allocate.

[0098]

(2) According to the above-mentioned (1), since unnecessary data transfer between the main memory and the cache memory can be reduced, power consumption along with data transfer is reduced and the congestion of data transfer routes is alleviated, improving throughput.



Initial address of a memory area  
to be released

Byte size of a memory to be released

$S+R$  (= Ending address of a memory area  
to be released)

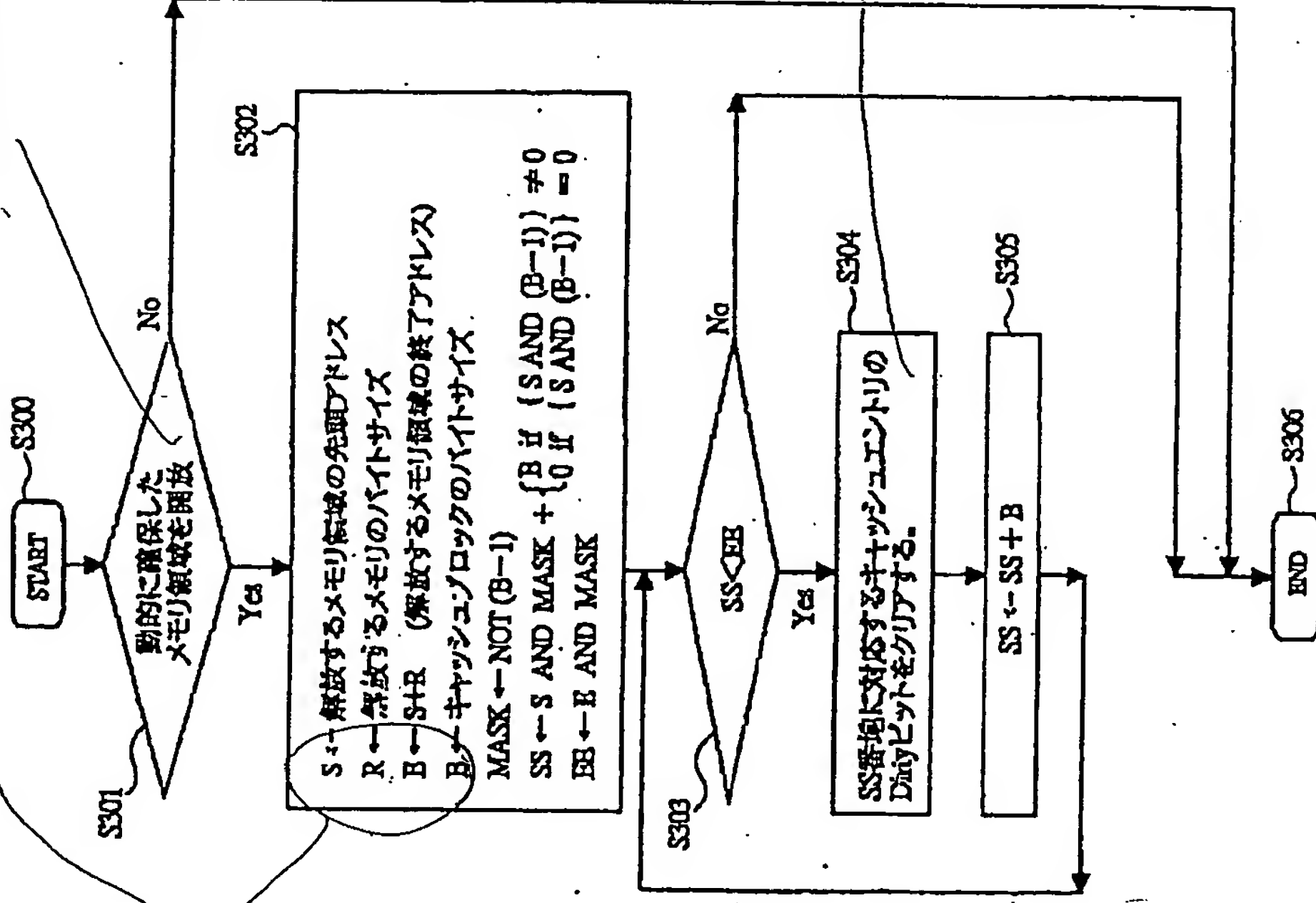
Cache block byte size

Figure 6

【図6】

図6

Releasing the dynamically reserved memory area



The cache entry Dirty bit  
corresponding to the address  
SS is cleared.

Figure 10

【O】

**10**

## RAM of Main Memory

257

## Bytes

四

serve

五

SP = 0x78DCB: A80 -

~~RE~~ Release

SP = 0.75000000

03

休

スタック用  
メモリ領域

## 主記憶装置のRAM

4x4t

010239567  
80 AUG 88

010239567  
80 AUG 88

010239567  
80 AUG 88

010239567  
80 AUG 88

Figure 7

【图7】

四



55.

0x1010

**Ex 100**

**0670150**

0x1040

國

—

Figure 8

Initial address of a memory area to be released

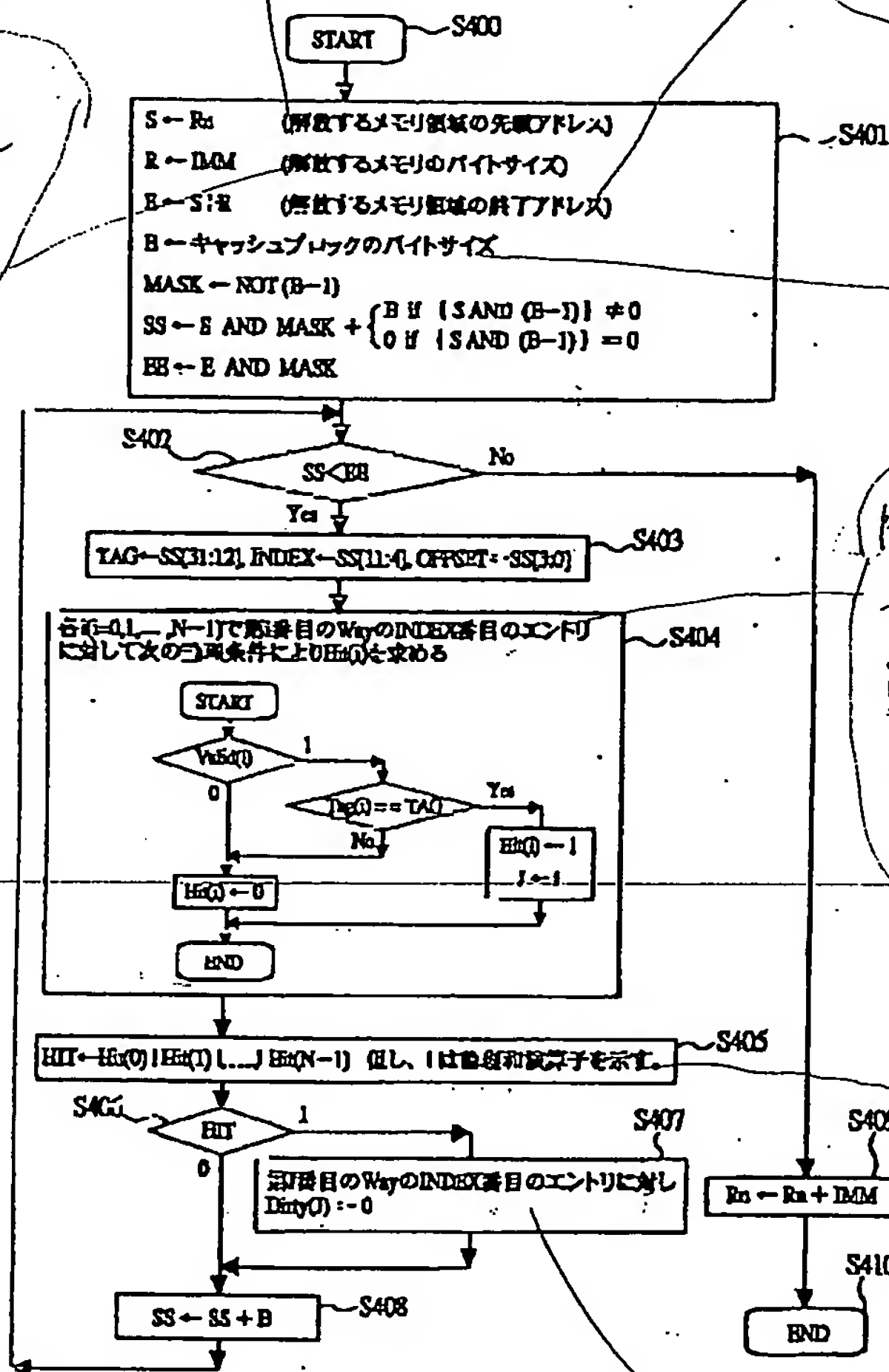
[ $\boxtimes$  8]

$\boxtimes$  8

Ending address of a memory area to be released

The Byte size of a memory to be released

The Byte size of a cache block



Hit(i) is calculated for the INDEX<sup>th</sup> entry of the i<sup>th</sup> way at each i (i = 0, 1, ..., N-1) according to the following logic requirement

I shows the logical OR operator

Dirty(J)

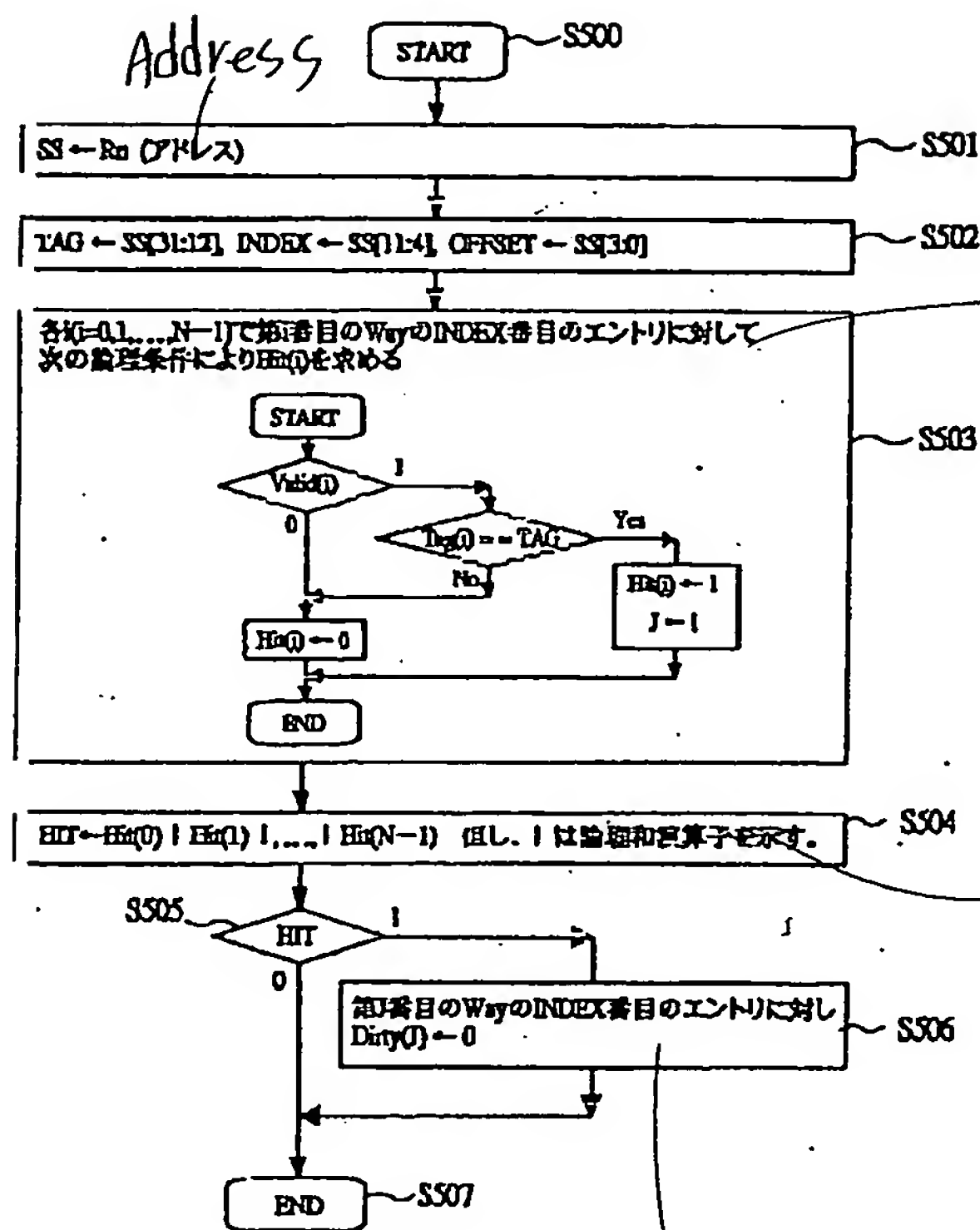
to the INDEX<sup>th</sup> entry of the J<sup>th</sup> Way

特 3) 103-223360 (P2003-223360A)

Figure 9

【図9】

図 9



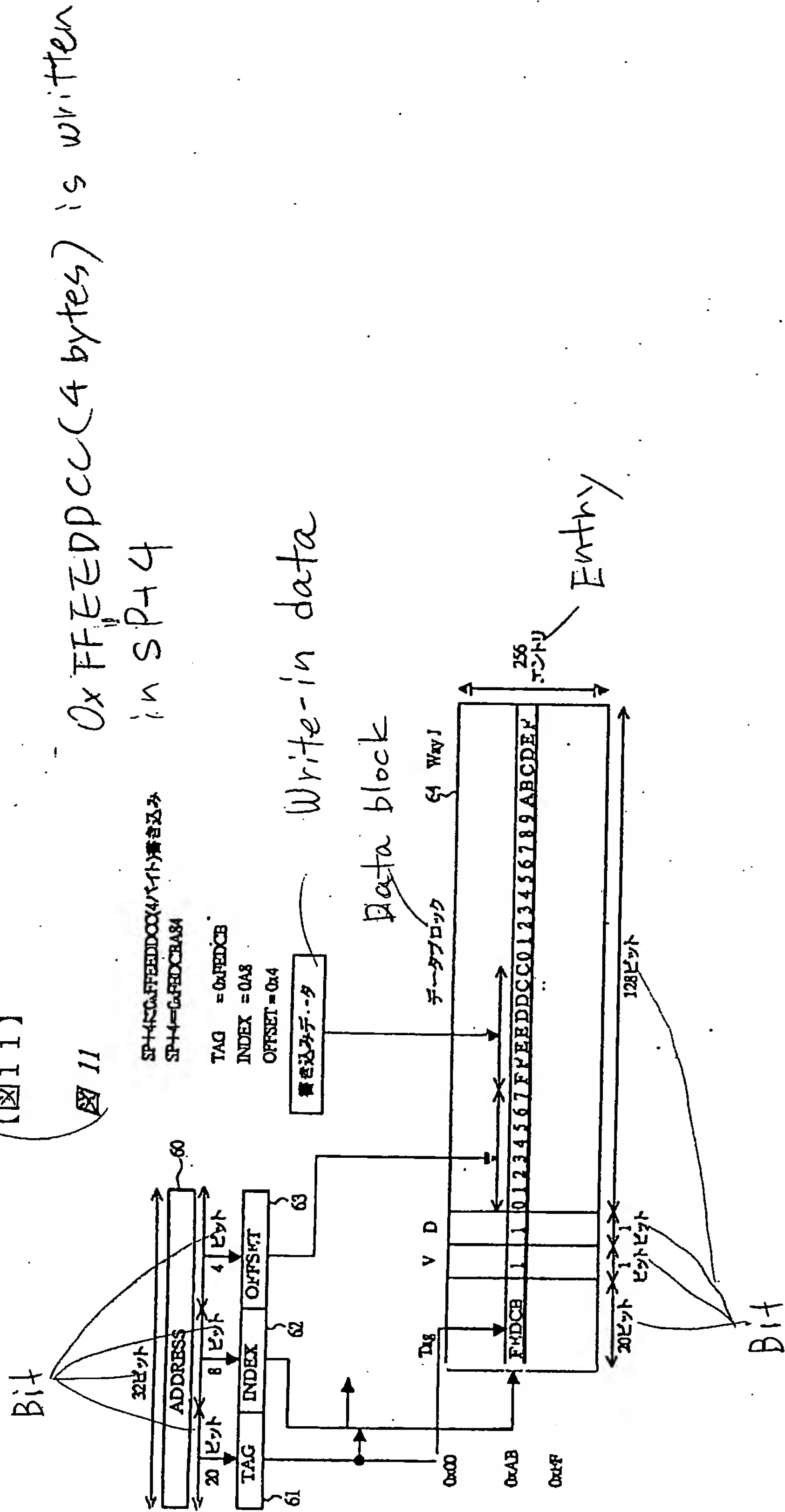
Hit(i) is calculated for the INDEX<sup>th</sup> entry of the i<sup>th</sup> Way of each i (i = 0, N-1) according to the following logic requirement

I shows the logical OR operator

Dirty(J) to the INDEX<sup>th</sup> entry of the J<sup>th</sup> Way



Figure 11



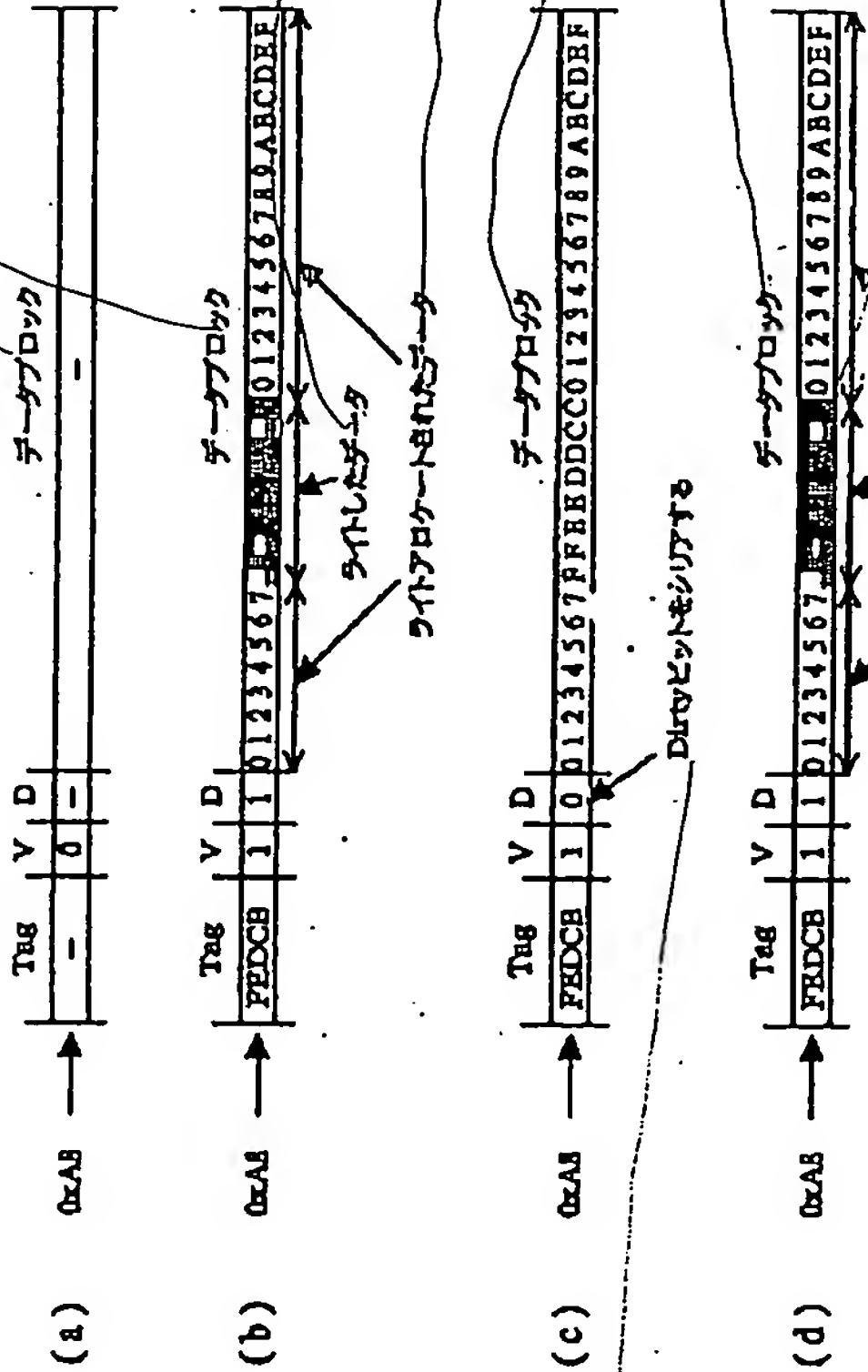
(註4) 103-223360 (P2003-223360A)

Figure 12

[12]

Data block

12



Write-in data

Write-allocate data

Data block

Data block

Write-in data

Data that is originally present in the data block

Dirty bit is cleared

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☒ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**